VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

INFORMAČNÍ SYSTÉM ORGANIZÁTORŮ ZÁVODŮ AGILITY

INFORMATION SYSTEM OF THE AGILITY COMPETITION ORGANIZERS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **MAGDALÉNA ONDRUŠKOVÁ**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. VLADIMÍR BARTÍK, Ph.D.**
SUPERVISOR

BRNO 2021

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2020/2021

Zadání bakalářské práce

Studentka: **Ondrušková Magdaléna**
Program: Informační technologie
Název: **Informační systém organizátorů závodů Agility**
Information System of the Agility Competition Organizers
Kategorie: Informační systémy

Zadání:

1. Seznamte se s principy tvorby webových aplikací, vývojovými prostředími a frameworky. Dále prostudujte problematiku získávání znalostí z databází.
2. Analyzujte požadavky na informační systém organizátorů závodů Agility (závody se psy) zahrnující možnost organizaci závodů, zadávání detailních výsledků, možnost registrace, platební bránu a propojení s Google kalendářem.
3. Prostudujte možnosti analýzy dat ze závodů (data mining, OLAP, popř. jiné) a navrhněte vhodnou úlohu pro analýzu těchto dat.
4. Navrhněte informační systém dle požadavků.
5. Navržený informační systém implementujte a otestujte jeho funkčnost na vhodném vzorku dat. Vyhodnoťte výsledky analýzy dat.
6. Zhodnoťte dosažené výsledky a další možnosti pokračování tohoto projektu.

Literatura:

- Žára, O.: JavaScript - Programátorské techniky a webové technologie, Computer Press, 2015. ISBN: 978-80-251-4573-9.
- Welling, L., Thomson, L.: PHP a MySQL: Kompletní průvodce vývojáře. CPress, 2017.

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 4.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Bartík Vladimír, Ing., Ph.D.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2020
Datum odevzdání: 12. května 2021
Datum schválení: 22. října 2020

Abstrakt

Cieľom tejto práce bolo navrhnúť a vytvoriť informačný systém pre organizátorov pretekov agility. Informačný systém je vytvorený ako webová aplikácia implementovaná v jazyku Python s využitím frameworku Django a databázovým systémom PostgreSQL. Podstatou práce bolo zjednodušiť organizovanie pretekov agility, poskytnúť pretekárom štatistiku výkonnosti psa ako aj analýzu výsledkov jednotlivých psov pomocou asociačných pravidiel. Užívateľom poskytuje možnosť prepojenia účtu s Google účtom, ako aj možnosť využívania Google kalendáru. Informačný systém bol úspešne vytvorený a otestovaný potenciálnymi používateľmi.

Abstract

The aim of this work was to design and create information system of the Agility Competition Organizers. The information system was created like web application implemented in Python language using Django framework and PostgreSQL database system. Final application simplifies an organization of agility competitions, provides competitors with statistics of dog and also analysis of results using association rules. Users can connect their account with Google account, and also can use Google calendar. Information system was successfully created and tested by potential users.

Kľúčové slová

informačný systém, webová aplikácia, agility, Python, Django, PostgreSQL, pretek, pretekári, organizátori pretekov, Google

Keywords

information system, web application, agility, Python, Django, PostgreSQL, competition, competitors, organization of competitions, Google

Citácia

ONDRUŠKOVÁ, Magdaléna. *Informační systém organizátorů závodů Agility.* Brno, 2021. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Vladimír Bartík, Ph.D.

Informační systém organizátorů závodů Agility

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracovala samostatne pod vedením pána Ing. Vladimíra Bartíka, Ph.D.Uviedla som všetky literárne pramene, publikácie a ďalšie zdroje, z ktorých som čerpala.

.....
Magdaléna Ondrušková
8. mája 2021

Poďakovanie

Tímto by som sa chcela poďakovať pánovi Ing. Vladimírovi Bartíkovi Ph.D. za odborné vedenie a pomoc pri tvorbe tejto bakalárskej práce. Takisto ďakujem mojej sestre Patrície a tatovi, ktorí ma priviedli k športu agility. Ďalej by som sa chcela poďakovať Natálii a všetkým pretekárom agility, ktorý mi poskytli potrebné informácie o problematike a za ich spätnú väzbu pri vývoji aplikácie.

Obsah

Kapitola 1

Úvod

Cieľom tejto práce je vytvoriť komplexný moderný informačný systém pre organizátorov pretekov agility ako aj pre samotných pretekárov. Agility je kynologický šport, kedy pes pod vedením psovoda prekonáva prekážky na parkúre, ktorý bol vytvorený rozhodcom pre daný beh. Informačný systém by mal organizátorom pretekov uľahčiť samotné organizovanie oficiálnych alebo neoficiálnych pretekov. Pretekárom by mal zároveň poskytnúť dobrú spätnú väzbu vo forme štatistiky, ale hlavne prehľadnejšie prihlasovanie sa na preteky. Návštevníkom stránky, ktorí ešte len rozmýšľajú, že začnú s agility, systém poskytne na základe precíznej analýzy dát pomocou asociačných pravidiel najvhodnejšie respektívne najúspešnejšie plemeno. Samozrejme, samotné plemeno nezaručuje úspech v pretekoch agility, ale daná analýza môže poskytnúť dobrú spätnú väzbu pri výbere nového psa.

V kapitole 2 sú dôkladnejšie vysvetlené princípy tvorby webových aplikácií, základné typy architektúr, ako je staršia klient-server a v súčasnej dobe viac používaná a preferovaná trojvrstvová architektúra. Ďalej sú v tejto kapitole predstavené základné jazyky a technológie, ktoré sú používané pri vývoji webových aplikácií. Na túto kapitolu nadväzuje kapitola 5, v ktorej sú podrobnejšie predstavené konkrétne použité technológie pre implementáciu informačného systému organizátorov agility - od technológií použitých na strane serveru ako je Python a jeho webový framework Django, cez databázovú vrstvu a konkrétny databázový systém PostgreSQL a samotnú prezentačnú vrstvu, teda konkrétne užívateľské rozhranie. V nasledujúcej kapitole 3 je vysvetlená problematika analýzy dát zo závodov pomocou dolovania dát a OLAP analýzy. V tejto kapitole sú vysvetlené základné metódy dolovania dát pomocou asociačných pravidiel, zhlukovania a klasifikácie.

Kapitola 4 vysvetľuje, čo je Agility a základné pravidlá, z ktorých vychádza základná funkcionalita informačného systému spolu so štatistikou pretekára. V tejto kapitole je zhodnotené aktuálne existujúce riešenie a to je porovnané s webovými aplikáciami existujúcimi v zahraničí. Na túto kapitolu nadväzuje kapitola 5, v ktorej sú rozobraté základné požiadavky organizátorov a pretekárov na daný informačný systém.

Použité technológie pre implementáciu navrhnutého informačného systému sú opísané v kapitole 6. Na túto kapitolu nadväzuje kapitola 7, v ktorej je rozobratý návrh aplikácie a kapitola 8, ktorá opisuje jej štruktúru a implementáciu jednotlivých častí. Neposlednou súčasťou práce bolo programové a užívateľské testovanie aplikácie, ktoré je opísané v kapitole 9. Poslednou súčasťou testovania bolo aj testovanie analýzy dát pomocou asociačných pravidiel.

V záverečnej kapitole 10 sú zhodnotené naplnenie hlavných cieľov práce ako aj vyhliadky do budúcnosti.

Kapitola 2

Princípy tvorby webových aplikácií

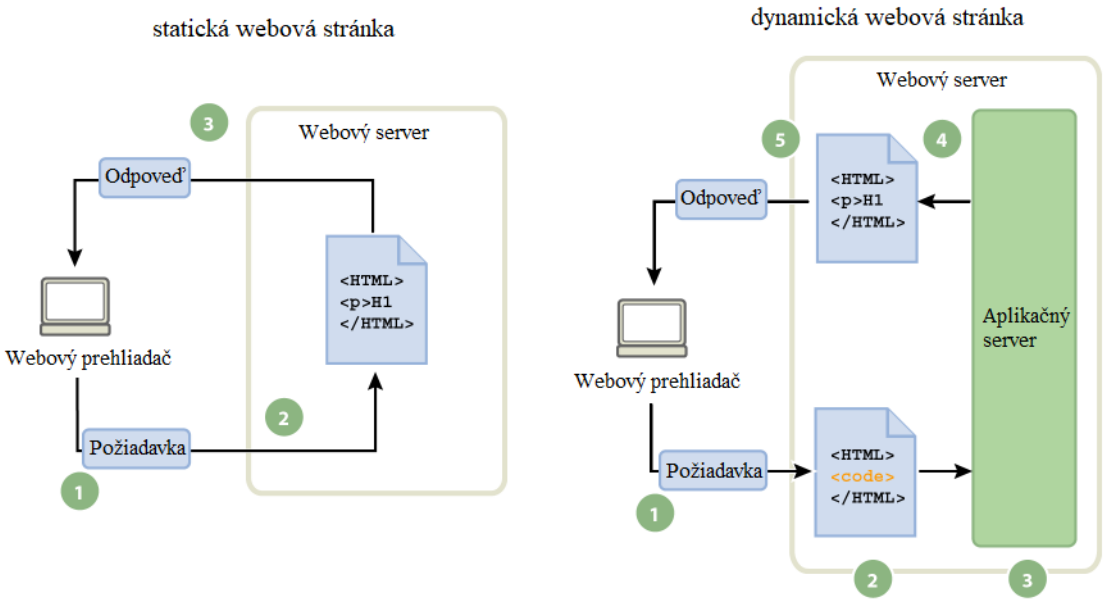
Kapitola *Princípy tvorby webových aplikácii* opisuje, čo je webová aplikácia a základné typy architektúr informačných systémov. V závere kapitoly sú predstavené a porovnané najpopulárnejšie jazyky a technológie pre vývoj webových aplikácií. Konkrétne technológie použité v rámci vývoja informačného systému sú opísané v kapitole 6.

2.1 Webová aplikácia

Webová aplikácia je aplikácia poskytovaná užívateľom z webového serveru pomocou internetu, ktorá nie je závislá na zariadení užívateľa a je spustiteľná z akéhokoľvek zariadenia pomocou webového prehliadača. Aplikácia beží na strane serveru, a preto sa webovému prehliadaču hovorí aj "tenký" klient (neobsahuje logiku aplikácie).[8]

Jednoduchosť a nenáročnosť na softvér je taktiež dôvodom, prečo sú v dnešnej dobe webové aplikácie medzi užívateľmi čoraz populárnejšie. Používajú sa napríklad pre implementáciu informačných systémov, internetových obchodov, rezervačných systémov a mnoho iného. Konkrétnymi príkladmi pre webové aplikácie sú napríklad: `www.spotify.com`, `www.starbucks.com`, `www.uber.com`.

Webová aplikácia je kombinácia statických a dynamických webových stránok. Statická webová stránka sa nemení, keď si ju užívateľ zobrazí - webový server odošle stránku webovému prehliadaču, ktorý o ňu požiadal, bezo zmeny. Každému užívateľovi sa táto stránka zobrazuje rovnako. Na rozdiel od statickej webovej stránky, dynamickú stránku webový server prenechá aplikačnému serveru, ktorý zodpovedá za dokončenie stránky. Aplikačný server prečíta kód na stránke a následne modifikuje stránku podľa inštrukcií v kóde. Výsledkom je statická stránka, ktorú aplikačný server odošle naspäť webovému serveru, ktorý ju odošle naspäť klientovi (webovému prehliadaču).[7] Princíp spracovania statickej a dynamickej stránky je ukázaný na obrázku 2.1. Príkladom dynamickej webovej stránky môže byť napríklad profil užívateľa, ktorý sa každému užívateľovi zobrazí podľa toho, aký užívateľ je prihlásený - napríklad iné meno, adresa, e-mailová adresa...



Obr. 2.1: Schéma spracovania statickej a dynamickej webovej stránky. Zdroj: [7]

Výhody webových aplikácií

- Aplikácia pracuje na webovom prehliadači, nezávisle od operačného systému.
- Užívateľ si nemusí nič inštalovať, ani danú aplikáciu aktualizovať (aktualizácie prebiehajú na strane serveru).
- Dáta sú uložené na strane servera, užívateľ k nim má prístup z akéhokoľvek zariadenia.

Nevýhody webových aplikácií

- Aplikácia vyžaduje internetové pripojenie, od jeho kvality sa odvíja rýchlosť toku dát a samotná práca s aplikáciou.
- Po vypnutí serveru, alebo ak sa poskytovateľ rozhodne aplikáciu ďalej neposkytovať, užívatelia k nej strácajú prístup a nemajú ďalej možnosť využívať túto aplikáciu, narozdiel od aplikácií, ktoré si užívateľ inštaluje na osobnom zariadení.
- Možné bezpečnostné riziká ako je napríklad únik dát v prípade nespoľahlivého poskytovateľa služby.

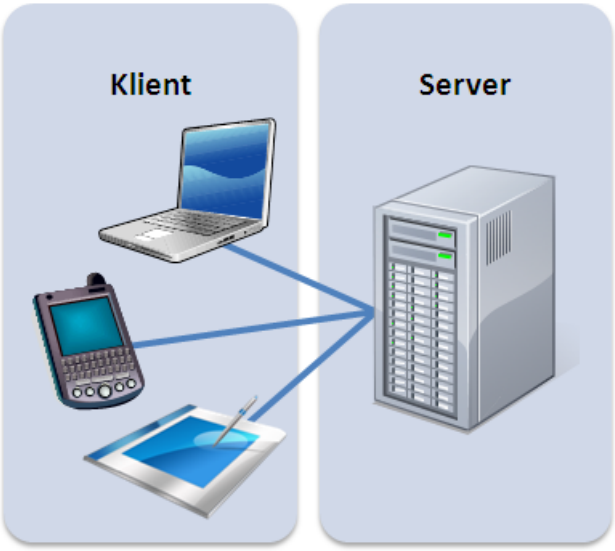
2.2 Architektúra

Webové aplikácie môžu byť štruktúrované rôzne. Architektúra popisuje spôsob rozdelenia jednotlivých častí informačného systému medzi serverovou a klientskou stranou. Existuje viacero druhov architektúr, z ktorých najznámejšie a najpoužívanejšie sú klient-server architektúra a trojvrstvová architektúra.

2.2.1 Klient-server architektúra

Architektúra klient-server je dvojvrstvová architektúra, ktorá je jedným z typov architektúr pre informačné systémy. Vrstvy architektúry, zobrazené na obrázku 2.2, sú: **klient** (aplikácia na zariadení užívateľa), ktorý zabezpečuje užívateľské rozhranie a aplikačnú logiku, a **server** (databáza), na ktorom sú ukladané dáta. Tieto vrstvy vzájomne po sieti komunikujú a vymieňajú si dáta: klient kladie požiadavky na server, server posiela klientovi odpovede (konkrétne dáta). Úlohou klienta je spracovať užívateľskú požiadavku tak, aby jej porozumel aj server, a následne, po obdržaní odpovede od serveru, spracovať túto odpoveď tak, aby bola zrozumiteľná pre užívateľa.[18]

Architektúra klient-server je už v dnešnej dobe zastaraná. Nároky na klienta sú vysoké, najmä pre aplikačnú logiku, ktorú obsahuje. Zložité je aj samotné aktualizovanie klienta. Nevýhodou je aj problematické sprístupnenie aplikácie pomocou webového prehliadača. Postupom času sa z dvojvrstvovej architektúry vyvinula trojvrstvová architektúra, ktorá je opísaná v podkapitole 2.2.2.



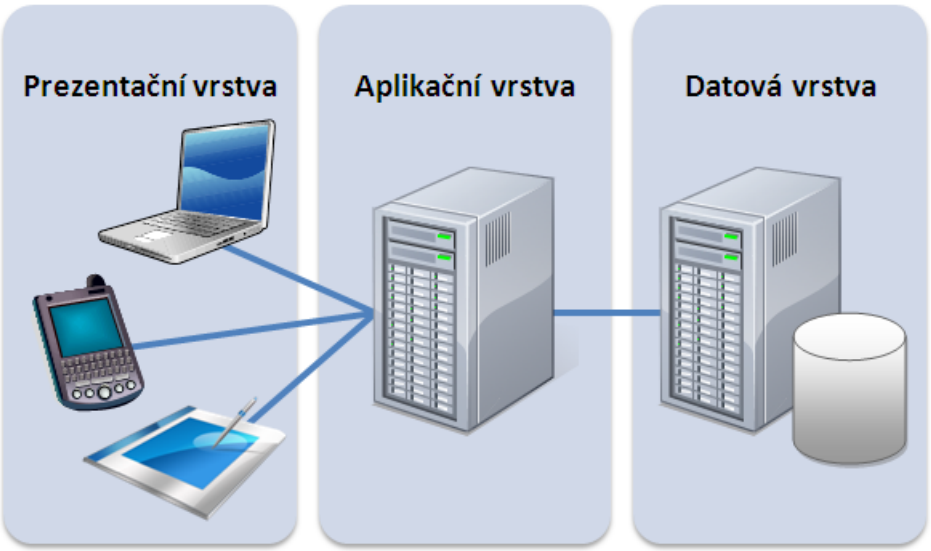
Obr. 2.2: Schéma architektúry klient-server. Zdroj: [18]

2.2.2 Trojvrstvová architektúra

Trojvrstvová architektúra, ukázaná na obrázku 2.3, je jedna z najviac používaných architektúr pre webové aplikácie, ktoré pracujú s veľkým množstvom dát. Skladá sa z troch častí: prezentačná vrstva, aplikačná vrstva a dátová vrstva. **Prezentačná vrstva** zobrazuje informácie užívateľovi, väčšinou formou grafického užívateľského rozhrania. Takisto prijíma požiadavky od užívateľa a môže ich aj kontrolovať - neobsahuje však samotné spracovanie dát.[5, sl. 8] Druhou vrstvou je **aplikačná** (funkčná) **vrstva**, ktorá je tvorená nástrojmi pre generovanie dynamických webových stránok. Obsahuje jadro aplikácie, logiku a funkcie, výpočty a spracovanie dát.[5, sl. 8] Poslednou vrstvou je vrstva **dátová** (databázová). Zaisťuje prácu s dátami (so systémom riadenia bázy dát) a základné dátovo-funkčné operácie, slúžiace na ukladanie, výber, agregáciu a integritu dát.[17]

Trojvrstvové a viacvrstvové architektúry využíva mnoho webových aplikácií. Ich hlavnou výhodou je oddelenie jednotlivých vrstiev tak, aby neboli na sebe vzájomne závislé -

vzniknutá aplikácia je robustná. Rozdeľuje výkon medzi zariadenie klienta a server. Požiadavky na zariadenie klienta sú menšie, a z tohto dôvodu môže prezentačná vrstva bežať aj na menej výkonných zariadeniach.



Obr. 2.3: Schéma trojvrstvovej architektúry. Zdroj: [17]

2.3 Jazyky a technológie

Mnoho webových aplikácií je napísaných priamo v čistých programovacích jazykoch ako je napríklad PHP. Dnes existuje ale mnoho aplikačných rámcov, takzvaných frameworkov, ktoré značne zjednodušujú a zautomatizovávajú vývoj aplikácie pre programátorov. Programátor má možnosť sa viac zamerať na vývoj samotnej aplikácie, jeho riešenie je abstraktnejšie a kód prehľadnejší. V nasledujúcej časti sú spomenuté jazyky a technológie, ktoré sú v dnešnej dobe najpoužívanejšie práve pre vývoj webových aplikácií.

2.3.1 HTML

HTML (*Hypertext Markup Language*) je značkovací jazyk používaný pre tvorbu webových stránok. Umožňuje vytvárať dokumenty obsahujúce text, hypertextové odkazy, multimediálny a iný obsah, formuláre, skripty a metainformácie zobraziteľné vo webovom prehliadači. Jednotlivé HTML elementy opisuje pomocou značiek (angl. tag) a ich vlastností (angl. attribute). Každý element je teda popísaný začiatočnou značkou, obsahom a ukončujúcou značkou, ktorá sa od začiatočnej značky líši iba lomkou, napríklad `<p>Obsah</p>` označuje odsek. Značky ale nemusia byť iba párové, napríklad `<hr>` vykreslí vodorovnú čiaru.[30]

2.3.2 CSS

CSS (*Cascading Style Sheets*) je jazyk pre opis spôsobu zobrazenia elementov na stránkach napísaných v HTML, XHTML a XML.[26] CSS poskytuje väčšie možnosti formátovania vzhľadu webovej stránky ako samotné HTML. Definuje zoznam pravidiel, ktoré sa skladajú zo selektoru a bloku deklarácií. Selektor určuje, na ktorý element dokumentu sa aplikujú

vlastnosti z bloku deklarácií. Blok deklarácií tvorí zoznam dvojíc - identifikátor vlastnosti a hodnota vlastnosti, oddelené bodkočiarkou. Pravidlá môžu určovať farbu pozadia stránky, veľkosť a font písma, zarovnanie a tak ďalej. Jednotlivé pravidlá môžu byť pre HTML dokument definované troma spôsobmi:[19]

- použitím atribútu `style` v HTML elemente;
- definíciou pravidiel v hlavičke dokumentu v elemente `<style>`;
- pripojením externého súboru pomocou elementu `<link>` v hlavičke dokumentu.

Pravidlá definované v rámci atribútu v konkrétnom HTML elemente platia iba pre daný element. Zvyšné dva spôsoby definujú pravidlá pre celý dokument a spravidla tu platí dedičnosť - napríklad definovaním fontu písma v pravidle pre značku `<body>`, definujeme font aj pre všetky elementy nachádzajúce sa v HTML dokumente medzi značkami `<body>` a `</body>`.[19] Písanie jednotlivých pravidiel pre všetky elementy v rámci webovej aplikácie môže byť zdĺhavé a náročné. Z tohto dôvodu vznikli CSS frameworky, ktoré poskytujú frond-end programátorom štýlopisy (angl. stylesheets) s vopred nadefinovanými pravidlami pre jednotlivé HTML elementy. Medzi najpoplárnejšie patria: Bootstrap, Tailwind CSS, Bulma a Foundation.[27]

2.3.3 JavaScript

JavaScript je multiplatformový, objektovo orientovaný skriptovací jazyk. Beží na strane klienta vo webovom prehliadači a je využívaný na ovládanie funkcionality webovej stránky - definuje správanie, ktoré nastane pri udalosti, ako je napríklad stlačenie tlačidla, vyplnenie poľa vo formulári, načítanie stránky a tak ďalej. Môže bežať ale aj na strane servera, pomocou multiplatformového prostredia Node.js.[21] Medzi najpopulárnejšie frameworky implementované v JavaScripte patria: React.js, Vue.js a Angular.js.[23]

Prácu s JavaScriptom uľahčuje knižnica jQuery[1], ktorá zjednodušuje manipuláciu s DOM (Document Object Model) elementami, volania AJAX a spracovanie udalostí. Viacero bežných úloh, pre ktoré vykonanie je treba dlhý a zložitý JavaScript kód, obaľuje do metód pre jednoduchšie použitie. Používa formát `$(selektor).akcia()`, kde selektor označuje element/y a akcia, označuje udalosť, ktorá nastane po vybraní selektoru, napríklad `("p").hide()` schová elementy so značkou `<p>`.[20]

Vyššie spomenuté technológie (HTML, CSS a JavaScript) sú používané najmä pre vývoj klientskej časti webovej aplikácie. V nasledujúcej časti tejto kapitoly sú preto porovnané najpopulárnejšie programátorské jazyky a frameworky implementované v týchto jazykoch, ktoré sú používané na vývoj serverovej časti aplikácie:[14]

- React.js, Angular.js, Vue.js - implementované v jazyku JavaScript;
- ASP.NET MVC - implementovaný v jazyku C#;
- Ruby on Rails - implementovaný v jazyku Ruby;
- Django, Flask - implementovaný v jazyku Python;
- Laravel, Symfony - implementovaný v jazyku PHP.

[1] https://jquery.com/

2.3.4 Laravel

Laravel[2] je voľne šíriteľný framework určený na zjednodušenie a zrýchlenie vývoja webových aplikácií pomocou vstavaných funkcií. Je založený na návrhovom vzore Model-View-Controller.[12] Medzi hlavné výhody Laravelu patrí integrácia s aplikáciami a najznámejšími e-mailovými službami prostredníctvom aplikačného programového rozhrania (API), kompatibilita s inými platformami a knižnicami tretích strán a automatizácia testovania pomocou PHPUnit. Medzi nevýhody patrí nekompatibilita naprieč verziami.[4]

2.3.5 Symfony

Podobne ako Laravel, aj Symfony[3] je voľne šíriteľný framework založený na návrhovom vzore Model-View-Controller. Hlavnou výhodou Symfony je veľká komunita vývojárov a kvalitne spracovaná a jednoducho pochopiteľná dokumentácia. Na rozdiel od Laravelu poskytuje podporu viacerým databázam (napríklad Oracle). Nevýhodou môže byť `Twig`, nástroj pre spracovanie šablón, ktorý v porovnaní s `Blade`, ktorý používa Laravel, neumožňuje znovuvyužiteľnosť kódu vrámci šablón.[4]

2.3.6 Flask

Flask[4] je voľne šíriteľný framework, určený na vývoj webov. Hlavnou výhodou Flasku je rýchlosť, jednoduchosť a integrácia s databázou. Poskytuje programátorovi dostatok nástrojov pre vývoj aplikácie a zároveň mu necháva dostatok kontroly a flexibility pre prispôsobenie si aplikácie podľa potrieb užívateľov. Ponúka viacero nástrojov pre prácu so šablónami, spolu s predvoleným `Jinja2`. Zároveň výhodou aj nevýhodou môže byť integrácia s databázou. Flask neposkytuje databázovú vrstvu a nespolieha sa na objektovo relačné mapovanie (angl. Object relational mapping, ďalej iba ORM). Na druhú stranu, integrácia s databázami typu NoSQL ako je napríklad MongoDB a DynamoDB, je veľmi jednoduchá. Autentizáciu, autorizáciu užívateľa alebo zmenu hesla musí programátor riešiť sám, napríklad pomocou rozšírenia, akým je napríklad `Flask-Security`.[11]

2.3.7 Django

Takisto ako Flask, Django[5] je voľne šíriteľný framework, ktorý sa používa pri vývoji webových aplikácií. Na rozdiel od Flasku poskytuje podporu aj pre autentizáciu užívateľa a ORM - umožňuje používanie databáz typu MySQL, PostgreSQL, SQLite. Je založený na návrhovom vzore Model-View-Template, ktorý vychádza z Model-View-Controller. Poskytuje administratívny panel, ktorý umožňuje programátorovi, prípadne správcovi systému jednoduchú manipuláciu s obsahom.[3] Na základe vyššie spomenutých vlastností bol práve Django framework vybraný pre implementáciu webovej aplikácií pre organizátorov agility a podrobnejšie bude popísaný v kapitole 6: Použité technológie.

[2] https://laravel.com/
[3] https://symfony.com/
[4] https://flask.palletsprojects.com/en/1.1.x/
[5] https://www.djangoproject.com/

Kapitola 3

Metódy analýzy dát

Táto kapitola je zameraná na stručné vysvetlenie problematiky analýzy dát pomocou dolovania dát a OLAP analýzy. Medzi základné metódy používané na získavanie znalostí z databázy, ktoré sú opísané v tejto kapitole patria: dolovanie asociačných pravidiel, zhlukovanie a klasifikácia. Pre samotnú analýzu dát zo závodov bola zvolená metóda dolovania asociačných pravidiel, u ktorej je vysvetlený aj základný algoritmus Apriori, ktorý bol neskôr implementovaný.

3.1 Dolovanie dát

Získavanie znalostí z databázy je dôležitá súčasť moderných informačných systémov. V rámci informačných systémov je uchovávané veľké množstvo dát, ktoré samotnému užívateľovi nemusí mať žiadnu výpovednú hodnotu. Dolovanie netriviálnych, skrytých, respektíve skôr nepoznaných a potenciálne užitočných modelov dát a vzorov z veľkého objemu dát reprezentuje získavanie znalostí z dát. Netriviálnosť znamená, že pre získanie informácií bolo nutné použiť komplikovanejší prístup ako iba kladenie SQL dotazov nad databázou. Skrytosť značí, že ide o modely a vzory, ktoré v dátach na prvý pohľad nie je vidieť. Potencionálna užitočnosť získaných znalostí hovorí o význame modelov a vzorov pre rozhodovanie.[31]

3.1.1 Proces získavania znalostí

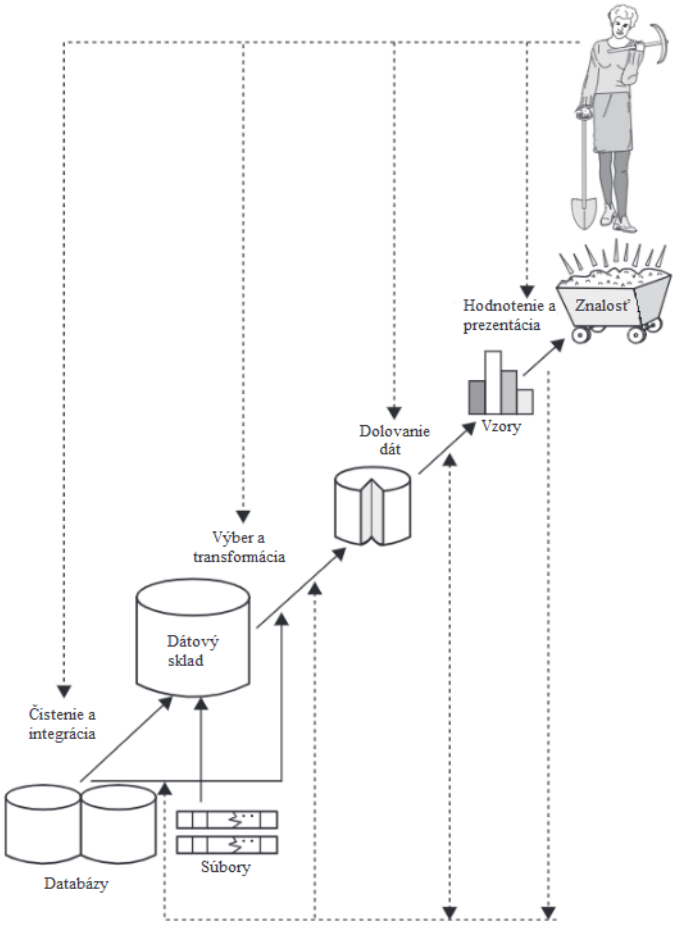
Je dôležité spomenúť, že dolovanie dát a získavanie znalostí z databázy sa v dnešnej dobe používajú ako synonymá, aj napriek tomu, že dolovanie dát je iba jedným krokom procesu získavania znalostí. Tento proces, znázornený na obrázku 3.1, prebieha vo viacerých krokov, ktoré sa iteračne opakujú:[13, Kapitola 1]

1. **Čistenie dát.** Účelom tohto kroku je zbavenie sa šumu a nekonzistentných dát v databáze.
2. **Integrácia dát.** V tomto kroku sú integrované dáta pochádzajúce z rôznych zdrojov. Častým riešením je spojiť prvé dva kroky a výsledné dáta uložiť do dátového skladu.
3. **Výber dát.** Z databázy sú vybrané dáta, ktoré sú relevantné pre úlohy na analýzu.
4. **Transformácia dát.** Cieľom tohto kroku je transformovať a zlúčiť dáta do foriem vhodných pre dolovanie dát. Niekedy sa tento krok uskutočňuje ešte skôr ako výber dát, najmä ak sa používa dátový sklad. Zároveň sa môže vykonávať aj **zjednodušenie dát**, za účelom vytvorenia menšieho objemu dát bez straty integrity dát.

5. **Dolovanie dát.** V rámci tohto kroku sú aplikované inteligentné metódy pre extrahovanie dátových vzorov.

6. **Hodnotenie vzorov.** V tomto kroku sú hodnotené vzory za účelom identifikácie skutočne zaujímavých znalostí.

7. **Prezentácia znalostí.** Cieľom tohto kroku je vizualizovať a prezentovať získané znalosti užívateľom.



Obr. 3.1: Proces získavania znalostí. Zdroj:[13, Obrázok 1.4]

V nasledujúcej časti kapitoly sú predstavené základné metódy dolovania dát a to konkrétne dolovanie dát pomocou asociačných pravidiel, zhlukovania a klasifikácie.

3.1.2 Dolovanie asociačných pravidiel

Úlohou asociačných pravidiel je nájsť zaujímavé asociácie alebo korelácie nad veľkými množinami dátových položiek. Základná definícia znie: Zvoľme si $\mathcal{I} = \{I_1, I_2, ..., I_m\}$ ako množinu položiek, D ako množinu transakcií, kde každá transakcia T je neprázdna množina položiek, u ktorej platí, že $T \subseteq \mathcal{I}$. Každá transakcia má svoj identifikátor, nazývaný TID. Ďalej si zvoľme A ako množinu položiek. Transakcia T obsahuje A, ak platí $A \subseteq T$. Asociačné pravidlo je implikácia $A \Rightarrow B$, kde $A \subset T$, $B \subset \mathcal{I}$, $A \neq \emptyset$, $B \neq \emptyset$ a $A \cap B = \emptyset$.

Je nutné si definovať ešte dva pojmy, a to konkrétne podporu a spoľahlivosť. Podpora X % znamená, že X % položiek, obsahuje obe množiny položiek A aj B. Spoľahlivosť Y % znamená, že v Y %, kde sa vyskytuje množina položiek A, sa vyskytuje aj množina položiek B. Matematická definícia znie pre oba pojmy nasledovne: asociačné pravidlo $A \Rightarrow B$ má **podporu** `s` v množine transakcií D, kde `s` je percento transakcií v D, ktoré obsahuje $A \cup B$. Asociačné pravidlo $A \Rightarrow B$ má **spoľahlivosť** `c` v množine transakcií D, kde `c` je percento transakcií v D, ktoré obsahuje A a tiež B. Tieto dva pojmy sa dajú pomocou pravdepodobnosti definovať nasledovne:

$$s(A \Rightarrow B) = P(A \cup B) \tag{3.1}$$

$$c(A \Rightarrow B) = P(B|A) \tag{3.2}$$

Pravidlá, ktoré spĺňajú zároveň minimálnu podporu *min_s* a minimálnu spoľahlivosť *min_c* sa nazývajú **silné**. Vo všeobecnosti sa dá získavanie asociačných pravidiel definovať pomocou dvoch krokov:[13, Kapitola 6]

1. **Nájdenie všetkých frekventovaných množín.** Všetky tieto frekventované množiny položiek sa musia vyskytovať aspoň tak často, ako vopred určená hranica pre minimálnu podporu *min_s*.

2. **Generovanie silných asociačných pravidiel z frekventovaných množín.** Tieto pravidlá musia spĺňať obe podmienky: podmienku pre minimálnu podporu *min_s* a podmienku pre minimálnu spoľahlivosť *min_c*.

Existuje viacero metód pre dolovanie asociačných pravidiel. Medzi najpoužívanejšie patrí: algoritmus Apriori a metódy FP stromu. U algoritmu Apriori je často problémom generovanie kandidátov, ktorých môže byť veľké množstvo. Ďalším problémom je nutnosť neustáleho prechádzania databázou. Jedným z možných riešení je metóda vzrastu frekventovaných množín. Prvou fázou tejto metódy je kompresia databázy reprezentujúca frekventované položky do štruktúry nazývanej FP-strom (strom frekventovaných množín). Tento strom sa potom rozdelí do podmienečných FP-stromov, ktoré sú vytvorené pre každú frekventovanú položku. Z týchto stromov sú potom získané frekventované množiny.[31, Kapitola 5].

Algoritmus Apriori

Algoritmus Apriori využíva pre získanie frekventovaných množín znalosť o skôr získaných frekventovaných množinách. V každej iterácií sú získané frekventované k-množiny použité pre vygenerovanie (k+1) množín. K zvýšeniu efektivity algoritmu sa používa takzvaná Apriori vlastnosť, ktorá hovorí o tom, že každá podmnožina frekventovanej množiny musí byť taktiež frekventovaná. Táto vlastnosť vychádza z faktu, že pridanie prvku do množiny nemôže spôsobiť, že vzrástla podpora množiny.[31, Kapitola 5]

Postup algoritmu Apriori je následovný:

1. V rámci prvej iterácie prejdeme celú databázu a pre každú položku databázy vypočítame jej podporu - položky predstavujú kandidátov pre frekventovanú množinu C_1.

2. Stanovíme si minimálnu podporu. Následne nájdeme frekventovanú množinu L_1 - túto množinu tvoria kandidáti C_1, ktorí spĺňajú minimálnu podporu.

3. Vygenerujeme kandidátov C_2 vytvorením každej možnej dvojkombinácií prvkov z množiny L_1, u ktorých nezáleží na poradí prvkov.

4. Vypočítame podporu kandidátov z C_2 a odstránime kandidátov, ktorí nespĺňajú podmienku minimálnej podpory - vznikne nám frekventovaná množina L_2.

5. Z frekventovanej množiny L_2 vygenerujeme kandidátov C_3 vytvorením trojprvkových množín. Ak niektorá z podmnožín vytvorených trojprvkových množín nie je frekventovaná v L_2, môžeme danú množinu odstrániť z kandidátov C_3.

6. Takto pokračujeme a zvyšujeme počet prvkov v množinách kandidátov C_k, až kým niektorá z množín C_k nie je prázdna - v tom prípade je možné algoritmus ukončiť a za výsledok sa považuje frekventovaná množina L_{k-1}

Pre vygenerovanie asociačných pravidiel z frekventovaných množín sa aplikuje rovnica pre výpočet spoľahlivosti:

$$c(A \Rightarrow B) = P(B|A) = \frac{s(A \cup B)}{s(A)} \tag{3.3}$$

Generovanie asociačných pravidiel prebieha v týchto dvoch krokoch:

- Pre každú frekventovanú množinu l, generuj všetky jej neprázdne podmnožiny.
- Pre každú podmnožinu s, vygeneruj pravidlo $s \Rightarrow (l - s)$ a podľa rovnice vypočítaj jeho spoľahlivosť. Ak je spoľahlivosť vyššia ako stanovená minimálna spoľahlivosť, potom je pravidlo silné.

Každé pravidlo generované z frekventovaných množín automaticky spĺňa podmienku minimálnej podpory. [31, Kapitola 5]

3.1.3 Zhluková analýza

Zhluková analýza, alebo zjednodušene zhlukovanie, je rozdeľovanie množiny dát do menších podmnožín. Každá podmnožina sa nazýva **kluster** a platí, že všetky objekty v jednom klustri sú si vzájomne podobné a zároveň sú odlišné od objektov v iných klustroch. Medzi najznámejšie metódy založené na rozdeľovaní patria k-means a k-medoids. Okrem týchto metód sa používa hierarchické zhlukovanie a zhlukovanie založené na hustote. Medzi menej používané metódy patria metódy založené na mriežke, metódy založené na modeloch a metódy pre zhlukovanie vysoko-dimenzionálnych dát.[31, Kapitola 7]

Metódy založené na rozdeľovaní

Tieto metódy rozdeľujú množinu n objektov do k tried, kde $k \leq n$, kde každá trieda k musí obsahovať aspoň jeden objekt. U týchto metód existuje podmienka, že objekt môže patriť práve do jednej triedy. Väčšina metód je založených na vzdialenosti. Princíp algoritmu spočíva v definovaní určitého množstva tried k a priradení prvotných objektov do týchto tried. Následne sú použité iteratívne relokačné techniky, ktorých cieľom je presúvať objekty z jednej triedy do druhej tak, aby objekty v jednej triede k si boli podobné a zároveň rozdielne od ostatných objektov v iných triedach. Najznámejšie algoritmy, ktoré využívajú metódy založené na rozdeľovaní sú **k-means** a **k-medoids**. Oba algoritmy pracujú na rovnakom princípe, líšia sa iba vo výbere prvkov - k-means vyberá prvky založené na centrálnom bode a k-medoids vyberá prvky na základe reprezentujúceho objektu.[13, Kapitola 10]

3.1.4 Klasifikácia a predikcia

Klasifikácia je forma analýzy dát, ktorá umožňuje priraďovať dáta na základe ich vlastností do konečného počtu tried. Táto analýza môže pomôcť užívateľom lepšie pochopiť všeobecné údaje. Predikcia je predpoveď istej hodnoty (obecne spojitého charakteru) pre daný objekt na základe jeho vlastností.[31, Kapitola 6]

Klasifikácia dát je dvojstupňový proces, ktorý zahrňuje **fázu učenia** a **fázu testovania klasifikátora**. V prvej fáze sú z databázy vybrané vzorky dát, ktoré tvoria takzvanú trénovaciu množinu. U týchto dát poznáme, do akej triedy sú zaradené. Klasifikátor, ktorého vstupom sú dáta z trénovacej množiny, sa pokúša zistiť takzvané **klasifikačné pravidlá**, pomocou ktorých je možné s určitou presnosťou priradiť konkrétny objekt do danej triedy. V druhej fáze prebieha testovanie klasifikátora pomocou takzvanej testovacej množiny. Testovaciu množinu tvoria dáta, u ktorých takisto poznáme triedu kam patria, ale zároveň sú to dáta ktoré by nemali byť obsiahnuté v trénovacej množine. Pomocou klasifikátora sú tieto dáta priradené do tried, pričom vďaka znalosti skutočnej triedy je možné vyhodnotiť percentuálnu úspešnosť klasifikátora, a jeho vhodnosť použitia na klasifikovanie dát, u ktorých nie je známe, do ktorej triedy patria.[31, Kapitola 6]

Medzi najznámejšie metódy klasifikácie patrí klasifikácia pomocou rozhodovacích stromov a Bayesovská klasifikácia. Klasifikácia pomocou rozhodovacích stromov je založená na vytvorení rozhodovacieho stromu, ktorý predstavuje graf stromovej štruktúry, kde každý vnútorný uzol reprezentuje test hodnoty istého atribútu a koncové uzly (listy) reprezentujú triedu, do ktorej je daný objekt klasifikovaný. Výhodou rozhodovacieho stromu je jednoduchosť prevodu na klasifikačné pravidlá. Narozdiel od klasifikácie pomocou rozhodovacích stromov, Bayesovská klasifikácia je založená na štatistike. Pre novú vzorku dát bude podľa štatistických metód určené, s akou pravdepodobnosťou patrí do jednotlivých tried. Vzorka je napokon zaradená do tej triedy, pre ktorú je najväčšia pravdepodobnosť, že sa v danej triede táto vzorka nachádza.[31, Kapitola 6]

3.2 Dátové sklady a OLAP analýza

Dátové sklady a OLAP analýza tvoria významnú časť pri získavaní znalostí z databáz. Tieto informácie môžu byť použité k detekcii podvodov, riadeniu výrobných procedúr a mnoho iného.

Dátové sklady (angl. data warehouses) zovšeobecňujú a zjednocujú dáta vo viacrozmernom priestore. Poskytujú architektúru a nástroje pre manažérov firiem pre systematické organizovanie, pochopenie a využitie dát pre uľahčenie rozhodovania v závažných otázkach. Jednoducho povedané, dátový sklad je dátové úložisko, ktoré je spravované oddelene od prevádzkových databáz organizácie. Konštrukcia dátových skladov zahŕňa čistenie dát, integráciu a transformáciu dát. Navyše, dátové sklady poskytujú nástroje online analytického spracovania (ďalej iba OLAP) na interaktívnu analýzu mnohorozmerných dát.

OLAP analýza vo všeobecnosti označuje proces dopytovania textových a číselných dát z dátových skladov pre analytické účely. Pomocou nej je manažér podniku schopný vytvárať rôzne scenáre pre skúmanie alternatívnych možností rozhodovania sa vo vedení a smerovaní firmy a vyhodnotiť ich vhodnosť a zmysel. Ďalej poskytuje možnosť previesť komplexné výpočty za účelom hľadania vzorov v dátach.[13, Kapitola 4] OLAP analýza je častou súčasťou takzvaných Business Intelligence aplikácií, ktoré predstavujú informačné systémy pre podporu plánovania v oblastiach podnikového manažmentu.

Kapitola 4

Agility

V nasledujúcej kapitole bude opísaný šport agility, jeho vznik a základné pravidlá pre organizáciu pretekov. V rámci tejto kapitoly je spomenutý už existujúci informačný systém pre organizáciu oficiálnych pretekov agility pre Slovenskú republiku a následne je porovnaný s už existujúcimi informačnými systémami v zahraničí a to konkrétne v Českej republike a v Rakúsku, kde sa pretekov agility zúčastňuje veľké množstvo slovenských pretekárov.

4.1 História agility

Agility ako šport vzniklo v sedemdesiatych rokoch dvadsiateho storočia v Anglicku. Prvé oficiálne agility skúšky sa konali v roku 1980. Z Anglicka sa agility rýchlo rozšírilo do celého sveta.[6, Kapitola 2]

Agility je určené pre akýkoľvek tím (dvojica pes a psovod), bez ohľadu na vek a pohlavie psa a psovoda. Pes bez obojka a vôdzky prekonáva na povel psovoda prekážky. Od psovoda sa vyžadujú rýchle reakcie a premyslené vedenie psa.[2]

Na Slovensku sa začalo agility rozvíjať začiatkom deväťdesiatich rokoch dvadsiateho storočia založením prvého klubu agility v Bratislave s názvom *Slovenský klub agility (SKA)*. Neskôr začali vznikať kluby agility po celom Slovensku, takzvané Miestne kluby agility (MKA). Po určitej dobe vznikla *Asociácia Slovenských Klubov Agility (ASKA)*, ktorá zastrešuje všetky kluby agility na Slovensku, stanovuje pravidlá pre registráciu nových klubov, skúšobný a súťažný poriadok agility a v neposlednom rade aj pravidlá organizácie pretekov agility.[1]

4.2 Pravidlá agility

Existuje niekoľko medzinárodných kynologických organizácií, z nich má každá svoje vlastné pravidlá. Každá národná organizácia má právo si pravidlá prispôsobiť. V základných pravidlách sa ale viac-menej nelíšia. Môže sa meniť výška prekážok, počet veľkostných alebo výkonnostných kategórií, prípadne sú zachované prekážky, ktoré v iných krajinách neexistujú alebo sa ďalej nepoužívajú.[6, Kapitola 3]. Preto je dôležité prihliadať na aktuálne platné pravidlá stanované národnou organizáciou, ktorou na Slovensku je ASKA.

Všeobecné pravidlá

Pravidlá spomenuté v tejto časti kapitole sú platné ako pre oficiálne, tak neoficiálne preteky s menšími rozdielmi a bolo nutné na ne prihliadať pri samotnom návrhu a implementácií informačného systému.

Účasť

Zúčastniť sa pretekov môžu psy všetkých plemien s preukazom pôvodu alebo bez neho, bez ohľadu na vek a pohlavie. Pre účasť na oficiálnych pretekov, tím (pes a psovod) musí mať vystavený výkonnostný zošit. Jeden psovod môže štartovať v jeden deň aj s viacerými psami. S jedným psom ale môžu v rámci rôznych pretekov pretekať viacerí psovodi, pokiaľ majú vystavený výkonnostný zošit. Pre účasť na neoficiálnych pretekov sa výkonnostný zošit od pretekárov nevyžaduje.[2]

Výškové kategórie

Existujú tri výškové kategórie, rozdelené na základe nameranej výšky psa v kohútiku: Small (S) pre psy s výškou menej než 35 cm, Medium (M) pre psy s výškou viac ako 35 cm a zároveň menšou ako 43 cm a Large (L) pre psy s výškou 43 cm a viac.[2]

Hodnotenie behov

Okrem štandardného hodnotenia behu podľa času sa rátajú aj trestné body. Trestné body môže pretekár získať za chybu, odmietnutie a prekročenie štandardného času. Za chyby a odmietnutia získava pretekár po 5 trestných bodov, za prekročenie štandardného času (daného rozhodcom) získava pretekár za každú stotinu 0,01 trestného bodu. Okrem trestných bodov môže byť pretekár diskvalifikovaný (napr. po prekročení maximálneho času, po získaní 3 odmietnutí, prekonávanie prekážok v nesprávnom poradí a podobne). Pri jednotlivých behoch Agility a Jumping sa používa následovné hodnotenie:[2]

- **Výborne:** 0 až 5,99 trestných bodov spolu.
- **Veľmi dobre:** 6 až 15,99 trestných bodov spolu.
- **Dobre:** 16 až 25,99 trestných bodov spolu.
- **Bez ohodnotenia:** 26 a viac trestných bodov spolu.
- **Diskvalifikácia**.

4.2.1 Oficiálne preteky

V rámci oficiálnych pretekov sa konajú viaceré typy súťaží. Existujú skúšky (rozdelené do výkonnostných kategórií) a otvorené preteky pre všetky tímy (Open Agility - OA, Open Jumping - OJ).

4.2.2 Organizátor

Organizátorom pretekov agility môže byť právnická i fyzická osoba, ktorá menuje garanta akcie. Organizátorom pretekov je preto vo väčšine prípadov klub agility. Garant akcie je organizátorom poverená osoba, ktorá zaisťuje organizáciu akcie od jej prípravy až po jej

vyhodnotenie. Organizátor má právo zmeniť termín, prípadne zrušiť akciu. Takisto má právo odmietnuť akúkoľvek prihlášku bez udania dôvodu.[2]

Výkonnostný zošit

Oficiálnych pretekov agility sa môžu zúčastniť len tie tímy, ktoré majú vystavený výkonnostný zošit. O jeho vystavenie žiada psovod výbor ASKA. Platnosť výkonnostného zošita začína dovŕšením osemnásteho mesiaca veku psa a po zapísaní kohútikovej výšky a výškovej kategórie psa.

Výkonnostný zošit je neprenosný a obsahuje nasledujúce údaje: meno a priezvisko psovoda, adresa bydliska, rok narodenia, príslušnosť klubu v ASKA, jednoznačné identifikačné číslo výkonnostného zošita, meno psa, plemeno, dátum narodenia psa, číslo mikročipu, tetovacie číslo psa, číslo SPKP, výšku v kohútiku psa, dátum merania a meno a podpis rozhodcu, ktorý dané meranie vykonal a dátum vydania.[2]

Výkonnostné kategórie

Existujú tri hlavné výkonnostné kategórie (A1, A2, A3). Každá kategória má inú náročnosť (napríklad A1 má ľahšie parkúry ako A2) a prestup medzi kategóriami je podmienený splnením určitých podmienok.

Výkonnostná kategória A1

Kategória je otvorená pre všetky tímy, ktoré majú vystavený výkonnostný zošit. Tímy, ktoré zabehli beh A1 trikrát s hodnotením výborne u dvoch rôznych rozhodcov na oficiálne uznaných pretekoch, môžu prestúpiť do výkonnostnej kategórie A2. Po prestupe do výkonnostnej kategórie A2 už tím nemôže štartovať vo výkonnostnej kategórii A1.[2]

Výkonnostná kategória A2

Môžu sa jej zúčastniť iba tímy, ktoré splnili podmienky prestupu z výkonnostnej kategórie A1 do A2. Pre postup z kategórie A2 do kategórie A3 je potrebné trikrát zabehnúť beh A2 s hodnotením výborne u minimálne dvoch rôznych rozhodcov. Tieto tri behy A2 musia byť bez trestných bodov, a zároveň sa tím v týchto behoch musí umiestniť do tretieho miesta.[2]

Výkonnostná kategória A3

Môžu sa jej zúčastniť iba tímy, ktoré splnili podmienky prestupu z výkonnostnej kategórie A2 do A3. Tímu je vydaná A3 karta, ktorá slúžia na zaznamenávanie bodov pre titul A3 šampióna. Aby pes dosiahol titul A3Champion musí získať sto bodov a minimálne 10x sa umiestniť na prvých troch miestach u minimálne dvoch rôznych rozhodcov. Body získava iba v prípade, že sa umiestnil do tretieho miesta (vrátane) bez trestných bodov. Počet bodov, ktoré tím získal, sa rovná počtu pretekárov, ktorých porazil. Tím, ktorému bola vydaná A3 karta, nesmie súťažiť v nižších výkonnostných kategóriách.[2]

4.2.3 Neoficiálne preteky

Neoficiálne preteky sa od tých oficiálnych líšia najmä v požiadavkách na organizovanie a podmienkach účasti. Organizovať ich môže ktokoľvek. Zúčastniť sa neoficiálnych pretekov

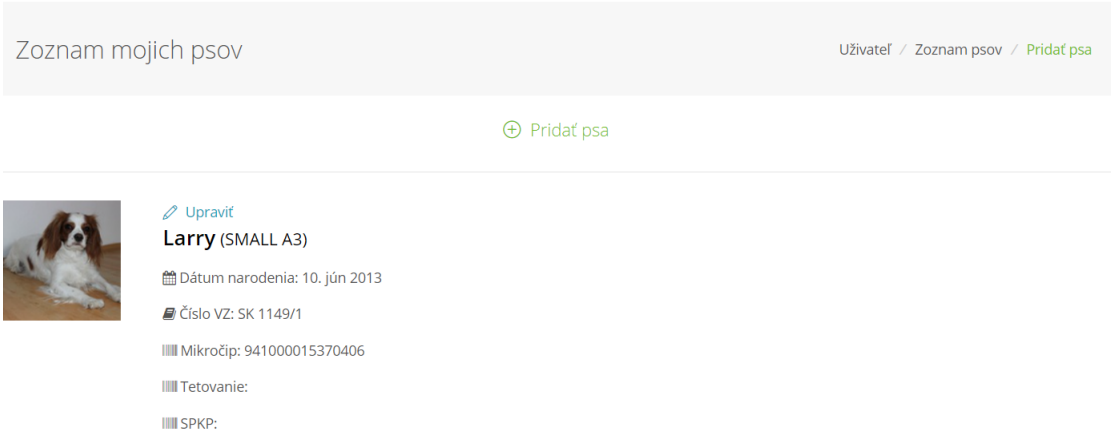
môže taktiež ktokoľvek, pretekár nemusí mať vystavený výkonnostný zošit. Hodnotenie behov je rovnaké ako v prípade oficiálnych pretekov.

4.3 Existujúce riešenie

Keďže agility sa na profesionálnej úrovni na Slovensku beháva už niekoľko rokov, v roku 2013 vznikol súčasný agility portál, www.agilityportal.sk/sk/. Agilityportal zastrešuje väčšinu klubov na Slovensku, ktoré organizujú oficiálne preteky. Možnosť organizácie neoficiálnych pretekov súčasný portál neumožňuje. Stránka prešla menšími zmenami začiatkom roku 2021, ale jedná sa skôr o zmeny vzhľadové ako funkčné.

Každý pretekár si vytvára užívateľský účet, v ktorom má pridaných psov, s ktorými tvorí tím. Ak s jedným psom tvorí tím viacero pretekárov, tento pes sa v systéme nachádza hneď niekoľkokrát duplicitne. Užívateľ nemá možnosť sa pozrieť na konkrétne výsledky psa zo všetkých pretekov bez ohľadu na pretekára, s ktorým pes bežal. Portál má aj ďalšie nedostatky: pretekár nevidí preteky, ktorých sa zúčastnil v minulosti a ani ich výsledky. Nemá k dispozícii ani základnú štatistiku výkonnosti psa. Užívateľ nevidí najnovšie pridané preteky na úvodnej stránke a nemá možnosť vyhľadať si konkrétne preteky, užívateľov, psov, prípadne klubov alebo rozhodcov. Takisto nemá možnosť si účet prepojiť s Google účtom, prípadne zaplatiť preteky organizátorovi.

Ako je vidieť nižšie na obrázku 4.1, užívateľ vidí iba základné informácie o psovi, ktoré tam sám nahral - nemá možnosť vidieť výsledky pretekov, ktorých sa daný pes zúčastnil, ani žiadnu štatistiku výkonnosti tohto psa. Túto stránku psa vidí iba prihlásený užívateľ, ktorý má daného psa nahratého v profile, neprihlásený alebo iný užívateľ nevidí ani tieto základné informácie o ľubovoľnom psovi.



Obr. 4.1: Zobrazenie psa na portále www.agilityportal.sk/sk/

4.4 Zahraničné portály agility

Oficiálne preteky Agility neexistujú iba na Slovensku, ale po celom svete. Každá krajina má svoje organizáciu, ktorá tento šport zastrešuje. V poslednej časti tejto kapitoly sú spomenuté dva informačné systémy existujúce v zahraničí, konkrétne v Českej republike a

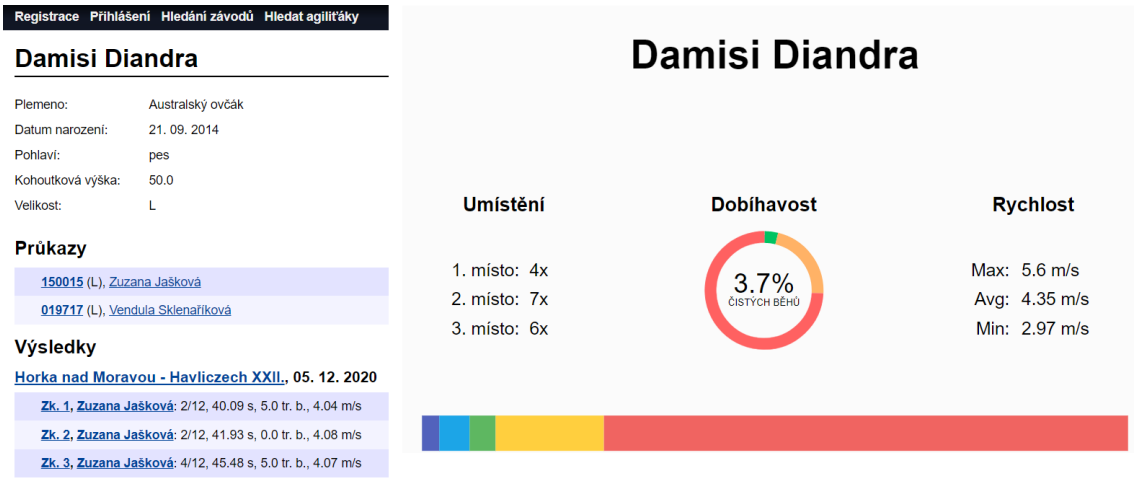
v Rakúsku. Tieto dva informačné systémy boli vybrané najmä vďaka blízkosti ku Slovensku a veľkej účasti slovenských pretekárov na pretekoch v týchto krajinách. Oba systémy sú komplexnejšie a poskytujú pretekárom viac funkcií v porovnaní s existujúcim riešením.

Česká republika: KAČR

Pre organizáciu pretekov agility v Českej republike existuje databáza agility www.kacr.info/. Užívatelia tohto systému môžu vyhľadávať (či už iných užívateľov, preteky alebo aj kluby), takisto si vedia pozrieť výsledky psov a minulých pretekov. Pretekár vidí prehľadne vo svojom profile jednotlivé behy a výsledok v rámci toho behu (umiestnenie, čas, trestné body, postupová rýchlosť...).

Na tento základný systém nadväzuje druhý, www.agistats.cz/, ktorý poskytuje detailnejšie prehľady výsledkov psa/psovoda a rôznu podrobnú štatistiku z pretekov. Súčasťou tejto stránky sú aj obecné štatistiky - prehľadná štatistiky vzhľadom na plemeno psa, štatistika rozhodcov a všeobecné štatistky pretekov.

Na obrázku 4.2 je vidieť zobrazenie konkrétneho psa na týchto dvoch portáloch. Vľavo vidíme všetky výkonnostné zošity s daným psom spolu s výsledkami daného psa, vpravo je možné vidieť zobrazenie toho istého psa spolu so základnou štatistikou jeho výkonnosti.



Obr. 4.2: Vľavo zobrazenie konkrétneho psa na www.kacr.info, vpravo zobrazenie toho istého psa na www.agistats.cz

Rakúsko: Dognow

V Rakúsku existuje webová aplikácia Dognow, www.dognow.net/ ktorá zahŕňa aj iné typy súťaží so psom ako agility. Užívateľom takisto poskytuje možnosť vyhľadania pretekov (podľa organizátora, dátumu, a iného), zobrazenia výsledkov pretekov a základnej štatistiky pretekára. Okrem týchto základných funkcií takisto zobrazuje mapu pretekov, pre užívateľa kalendár s prihlásenými pretekmi, pri jednotlivých pretekoch poskytuje možnosť užívateľovi poslať správu organizátorovi, prehľadné filtre pre zobrazenie už prihlásených/registrovaných pretekárov na preteky a tak ďalej.

Kapitola 5

Analýza požiadavok

V tejto kapitole sú definované požiadavky organizátorov, pretekárov ale aj neregistrovaných užívateľov na informačný systém. Požiadavky na systém boli vytvorené na základe dotazníka a na základe rozhovorov s viacerými pretekármi agility, ktorí existujúci agilityportal používajú. Dotazník bol zdieľaný hlavne na Facebooku prostredníctvom skupiny Agility komunita SR. Prebiehal najmä v mesiacoch október a november a zúčastnilo sa ho 56 ľudí, naprieč všetkými vekovými kategóriami. Na základe dotazníka je jasné, že skoro všetci opýtaní doteraz využívajú existujúci agility portál.

5.1 Organizátor agility

Z 56 ľudí, ktorí sa zúčastnili dotazníka, patrilo 12 k organizátorom pretekov agility. 8 ľudí zvolilo možnosť, že organizujú oficiálne preteky, 1 človek organizuje neoficiálne preteky a 3 organizujú aj oficiálne aj neoficiálne preteky. Z dotazníka vyplýva, že organizátori by najviac privítali tieto funkcie:

- **Spätná väzba po pretekoch.** Viac ako polovica organizátorov by privítala možnosť spätnej väzby od užívateľov (pretekárov) k pretekom.
- **Nahrávanie výsledkov po pretekoch.** Organizátori by privítali možnosť nahratia výsledkov pretekov do systému.
- **Možnosť kontaktovať pretekárov.** Polovica organizátorov by mala záujem o možnosť kontaktovania pretekárov.

Základná funkcionalita, ktorou je najmä pridanie samotných pretekov a úprava informácií, prípadne zrušenie pretekov, sa v dotazníku nenachádzala, je implicitná pre organizátora a automaticky sa s ňou počítalo pri vývoji informačného systému.

5.2 Pretekár agility

Na zvyšné otázky z dotazníku mohli odpovedať aj organizátori (keďže často sú aj oni sami pretekármi). Z dotazníka vyplýva, že väčšina užívateľov hodnotí aktuálny agilityportál ako dostačujúci s určitými výhradami. Najčastejší problém bol, že stránka je častokrát nefunkčná, neprehľadná, prihlasovanie na preteky je zložité, informácie sú neúplné a chýbajú základné funkcionality (elektronický výkonnostný zošit, výsledky behov a iné).

V dotazníku ľudia odpovedali na viacero otázok ohľadom funkcionality systému. Na hlavnej stránke by privítali najmä tieto nové funkcie:

- **Vyhľadávanie.** Až 77 % ľudí by privítalo vyhľadávanie pretekov/pretekárov/psov.
- **Najnovšie pridané preteky.** Až 65 % užívateľov by v rámci hlavnej stránky chcelo vidieť aj naposledy pridané preteky. Pretekov sa na Slovensku síce neorganizuje veľa, ale je dôležité, aby užívatelia videli nové preteky a žiadne pri rozhodovaní prihlásiť sa neprehliadli.
- **Mapa pretekov.** Až 60 % opýtaných by privítalo, keby si v rámci webovej aplikácii vedeli zobraziť mapu Slovenskej republiky spolu s vyznačením miestom konania nadchádzajúcich pretekov.

Vyššie vymenované nové funkcie by mali dopĺňať už existujúcu funkcionalitu, ako je zobrazenie nadchádzajúcich pretekov a klubov.
V ďalšej otázke respondenti vyberali informácie, ktoré považujú za dôležité a mali by sa nachádzať pri každých pretekoch. Nasledujúce informácie boli vybrané aspoň polovicou respondentov: základné informácie (dátum konania, miesto konania - adresa a GPS), výška štartovného (a ako sa mení vzhľadom na počet psov), kto je organizátorom a garantom akcie, kto je rozhodca (rozhodcov môže byť viac), program pretekov (v deň konania, čas začiatku jednotlivých kategórií a behov) a doplňujúce dokumenty (propozície). Pri mieste konania by pretekári ocenili, ak by bol spomenutý terén, na ktorom sa preteká (tráva/umelá tráva/piesok), prípadne či sa preteká v hale alebo vonku. Dôležitou informáciou je určite aj o aký typ pretekov ide (open/skúšky/neoficiálne preteky), prípadne aké typy jednotlivých behov sú organizované (v rámci jedných pretekov môžu byť napr. otvorené behy aj skúšky). Zo štatistických údajov pri zobrazení pretekov by užívatelia privítali prehľadné zobrazenie prihlásených a zaplatených pretekárov podľa výškovej kategórie (S/M/L) a výkonnostnej kategórie (A1/A2/A3). Po uskutočnení pretekov by 79 % respondentov privítalo zobrazenie výsledkov.
Následne opýtaní odpovedali v dotazníku na nové funkcie k prihláseným pretekom:

- **Platba za preteky.** Skoro 80 % pretekárov by privítalo možnosť zaplatiť preteky.
- **Kontaktovanie organizátora.** Skoro polovica opýtaných by privítala mať možnosť priamo kontaktovať organizátora, napríklad z dôvodu poslania žiadosti o platbu na mieste.
- **Prepojenie s Google účtom.** Tretina respondentov by privítala možnosť mať účet prepojený priamo s Google a využívať jeho funkcionalitu - najmä kalendár pre zobrazenie pretekov.
- **Zmena kategórie.** Pri zmene údajov o psovi, konkrétne jeho výkonnostnej kategórii, by niektorí opýtaní privítali, aby sa automaticky zmenila aj kategória v už prihlásených pretekoch.

Viacero užívateľov malo problém pri aktuálnom agilityportále s tým, že je veľmi neprehľadný. Často, keď bol užívateľ nový, nevedel ako vybrať psa s ktorým sa registruje na preteky (nutné kliknúť na fotku psa), nemal možnosť vidieť prehľadne vyfiltrované tímy danej výškovej alebo výkonnostnej kategórie.
Posledná časť dotazníka bola zameraná najmä na štatistické údaje z pretekov, ktoré zaujímajú jednotlivých pretekárov. Najčastejšie pretekárov zaujímalo:

- **Štatistika umiestnenia.** Až 84 % respondentov by zaujímala štatistika umiestnenia - počet koľkokrát sa daný pes umiestnil na danom mieste (prvé/druhé/tretie miesto), prípadne rozmedzie miest (do piateho miesta).
- **Rýchlosť.** Takmer 80 % respondentov by privítalo aj prehľadnú štatistiku rýchlosti psa - jeho priemernú rýchlosť, maximálnu rýchlosť, najmenšiu rýchlosť.
- **Dobiehavosť.** Viac ako 75 % opýtaných zaujíma štatistika dobiehavosti - koľko pretekov daný pes úspešne dobehol a nebol diskvalifikovaný.
- **Trestné body.** Takmer 60 % pretekárov by zaujímala štatistika trestných bodov - koľko trestných bodov bolo za chyby, koľko za odmietnutia a koľko za čas.
- **Body na A3Champion.** Cieľom každého pretekára je získať titul A3 šampión. Na to ale potrebuje nazbierať sto bodov. Viac ako 68 % respondentov by preto privítalo keby mohli v rámci štatistiky vidieť, koľko bodov získali na A3 šampióna.

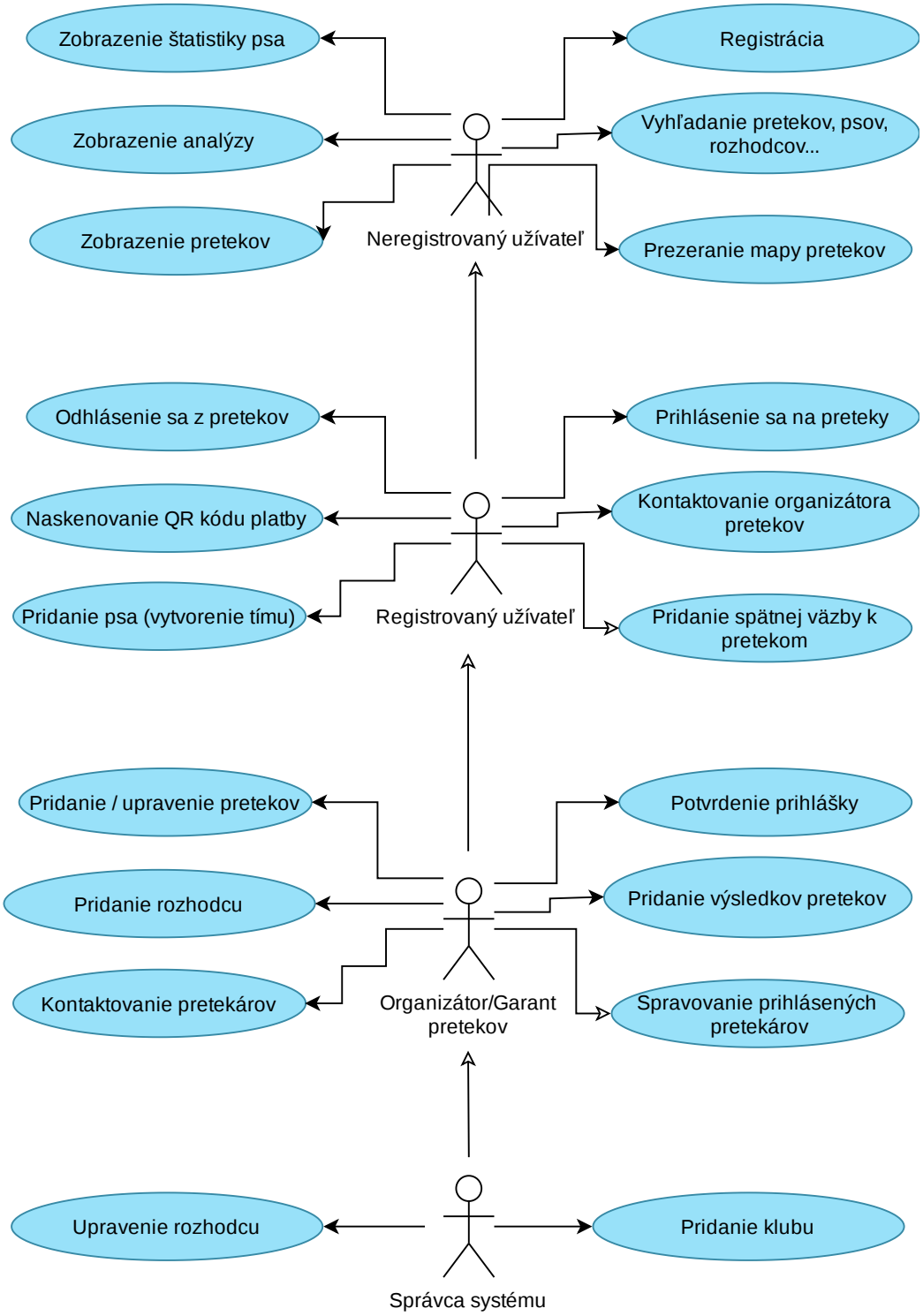
Všetky štatistiky vyššie spomenuté budú úplné len v prípade, že pretekár sa nezúčastňuje na žiadnych pretekoch v zahraničí. Ak sa pretekár takýchto pretekov zúčastňuje, tieto preteky nie sú vrámci informačného systému spracovávané a do štatistiky psa sa nezarátávajú. Body na A3Championa sa rátajú takisto aj zo zahraničia. Ak pretekár dosiahne tento titul aj vďaka bodom zo zahraničia, titul mu vrámci systému vie prideliť len správca systému.

5.3 Neregistrovaný užívateľ

Neregistrovaný užívateľ bude mať možnosť prezerania si pretekov, prípadne mapy pretekov, vyhľadávania pretekov/psa/rozhodcov a tak ďalej. Takisto si môže prezerať štatistiku jednotlivých psov ako je štatistika jeho umiestnenia, dobiehavosti a zobrazenie mnohých ďalších zaujímavých faktov ako je počet nazbieraných trestných bodov, čas strávený na parkúre, počet prekonaných prekážok a mnoho iného. Takisto si neregistrovaný užívateľ môže zobraziť analýzu najúspešnejších plemien, v rámci ktorej vidí kombináciu vlastností psov, ktoré sa najčastejšie umiestňovali na prvých 5 miestach.

5.4 Diagram prípadov použitia

Na základe analýzy požiadavkov popísaných v tejto kapitole bol vytvorený diagram prípadov použitia zobrazujúci typy užívateľov informačného systému a ich základnú funkcionalitu. Vytvorený diagram je znázornený na obrázku 5.1.



Obr. 5.1: Diagram prípadov použitia

Kapitola 6

Použité technológie

V nasledujúcej kapitole sú opísané použité technológie pre vývoj informačného systému. V rámci kapitoly je predstavený základný návrhový vzor softvéru, Model-View-Controller a Model-View-Template, ktorý z neho vychádza. Ďalej sú spomenuté a opísané technológie použité na vývoj serverovej časti (back-end), databázovej vrstvy a klientskej časti (front-end).

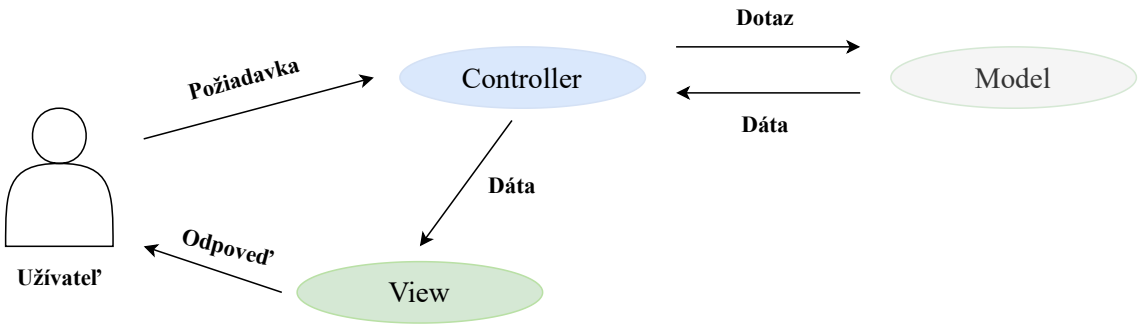
6.1 Architektonický vzor

Architektonický vzor obecne definuje rozloženie zložitejšej aplikácie do menších logických celkov, ktoré spolu vzájomne komunikujú. Ich hlavným cieľom je pomôcť vývojárovi pri návrhu aplikácie.

6.1.1 Model-View-Controller

Model-View-Controller, ďalej iba MVC, je architektúra, ktorá rozdeľuje aplikáciu do troch nezávislých častí - dátový model aplikácie (model), užívateľské rozhranie - pohľad (view) a riadiaca logika - kontroler (controller), tak, že modifikácia jednej časti má len minimálny vplyv na časti ostatné.

Ako je možné vidieť na obrázku 6.1, hlavnou úlohou modelu je správa dát - či už samotných údajov, ktoré sa prenášajú medzi pohľadom a riadiacou logikou, alebo inú logiku týkajúcu sa spracovania dát. Úlohou pohľadu je spracovanie dát pre zobrazenie užívateľovi - vytvorenie užívateľského rozhrania, s ktorým môže užívateľ pracovať. Kontroler funguje ako rozhranie medzi pohľadom a modelom, manipuluje s modelmi a renderuje pohľady.[28]

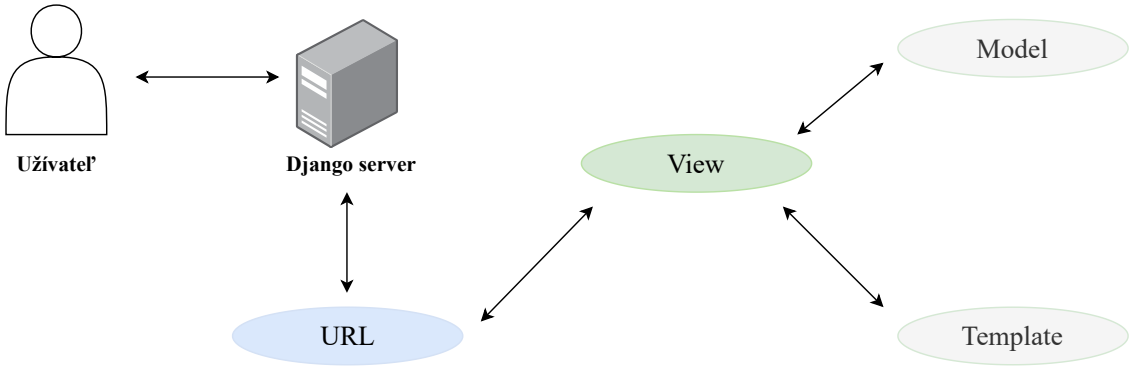


Obr. 6.1: Schéma MVC architektúry

6.1.2 Model-View-Template

Model-View-Template, ďalej iba MVT, je podobná architektúra ako MVC, ktorej sa drží framework Django. Na rozdiel ale od MVC, MVT neobsahuje kontroler - jeho funkciu plní samotné Django. Má takisto tri nezávislé časti - model, pohľad (angl. view) a šablóna (angl. template). Na obrázku 6.2 môžeme vidieť schému tejto architektúry a ako jednotlivé časti medzi sebou spolupracujú.

Model, podobne ako pri MVC, spravuje dáta a obsahuje logickú štruktúru celej aplikácie. Dáta sú reprezentované v databáze, ako je napr. MySQL, PostgreSQL a tak ďalej. Pohľad, na rozdiel od pohľadu v MVC, obsahuje biznis logiku, pomocou ktorej spolupracuje s modelom a renderuje šablóny, prijíma HTTP požiadavky (angl. HTTP requests) a odosiela HTTP odpovede (angl. HTTP responses). Najväčším rozdielom medzi MVC a MVT architektúrou je práve šablóna a mapovanie URL adries na príslušné pohľady, ktoré vykonáva samotný server. Šablóna sa správa ako prezentačná vrstva, je to prakticky HTML kód, ktorý renderuje dáta. Obsah týchto súborov môže byť bud statický, alebo dynamický.[9]



Obr. 6.2: Schéma MVT architektúry

6.2 Back-end

Back-end, alebo serverová časť, je samotné jadro aplikácie, ktorá definuje funkcionalitu webovej stránky. Pri webových aplikáciach beží väčšinou na strane serveru, ale môže bežať aj na klientskej strane - ak je klient 'hrubý". Informačný systém bol implementovaný vo frameworku Django a celá logika aplikácie sa nachádza práve na strane serveru - zariadenia užívateľov systému predstavujú "tenkého" klienta. Na serverovej strane beží aj databáza, konkrétne bola použitá databáza PostgreSQL, ktorá je popísaná v ďalšej podkapitole.

6.2.1 Python

Python je vysokoúrovňový, skriptovací, objektovo orientovaný programovací jazyk ktorý má široké použitie. Používa sa napríklad na: vývoj webu (serverová časť), vývoj softvéru, rôzne matematické výpočty a analýza dát a iné.[25]

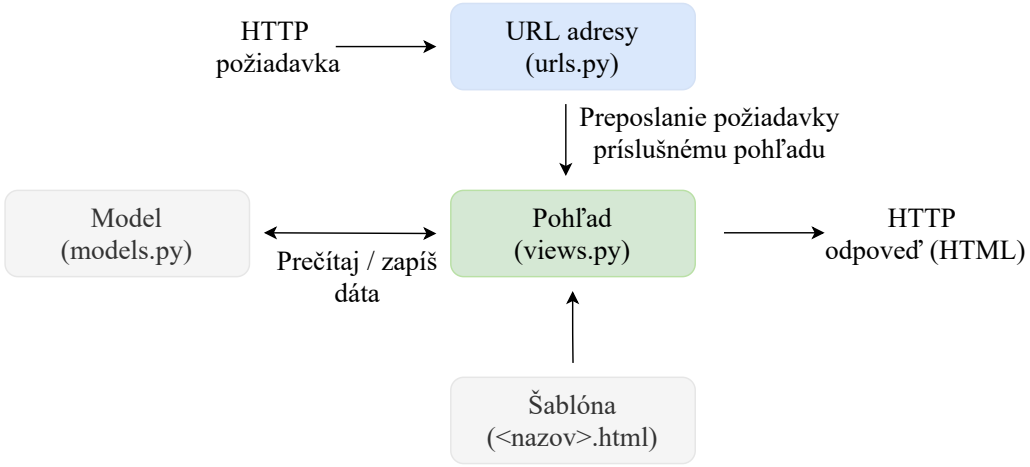
Python má viacero verzií, staršia, Python 2, je nekombalititná s verziou 3, a od začiatku roka 2020 už nie je ďalej oficiálne podporovaná. V rámci práce je preto použitý Python 3, konkrétne verzia 3.8.8, ktorej plánovaná podpora má skončiť v roku 2024. Kompatibilita naprieč podverziami (napríklad 3.7 a 3.8) býva väčšinou zaručená a bezproblémová.

Medzi najpopulárnejšie frameworky pre tvorbu webových aplikácií patria: Django, TurboGears, Flask a Bottle.[24]

6.2.2 Django

Django je vysokoúrovňový webový framework implementovaný v jazyku Python, ktorý umožňuje rýchly vývoj bezpečných a udržateľných webových aplikácií. Je založený na architektúre MVT. Obsahuje objektovo relačný mapovač (angl. object-relational mapper), ktorý sprostredkováva komunikáciu medzi dátovými modelmi (definovanými ako triedy v jazyku Python) a relačnou databázou.[22]

Na obrázku 6.3 je znázornené spracovanie HTTP požiadavku pomocou frameworku Django. Daná HTTP požiadavka je preposlaná príslušnému pohľadu (definovaný podľa URL adresy v súbore `urls.py`), ktorý požiadavku spracuje a posiela HTTP odpoveď užívateľovi.



Obr. 6.3: Spracovanie HTTP požiadavky Djangom.

Django ako framework poskytuje napríklad tieto komponenty:

- serializačný a validačný systém pre formuláre, ktorý automaticky prekladá dáta medzi HTML formulárom a hodnotami vyhovujúcimi databáze;
- šablónový systém, ktorý používa dedičnosť (podobne ako pri objektovo orientovanom programovaní);
- interný komunikačný systém, ktorý umožňuje komponentom aplikácie vzájomne komunikovať prostredníctvom preddefinovaných signálov;
- internacionálny systém, vrátane prekladov Django komponentov do rôznych jazykov;
- rozhranie pre jednotkové testovanie.

6.2.3 PostgreSQL

PostgreSQL[1], takisto známe ako Postgres, je objektovo relačný databázový systém, ktorý je použitý pre ukladanie dát na serverovej strane o užívateľoch (pretekárov a organizátorov),

[1] https://www.postgresql.org/

o psoch a samotných pretekoch. Je to voľne šíriteľný softvér, ktorý postupne vyvíja skupina programátorov po celom svete. Pre potreby práce bola použitá najnovšia verzia - 13.2. Oproti iným databázovým systémom (ako je napr. MySQL, MariaDB a tak ďalej), poskytuje navyše aj iné funkcie a dátové typy - formát JSON, ktorý PostgreSQL spracováva ako stĺpec (a pristupuje k nemu pomocou indexu), XML dátový typ, UUID pre lepšie spracovanie ID v rámci tabuľky a mnoho ďalšieho. Na rozdiel od MySQL poskytuje užívateľovi možnosť uložiť dátum a čas aj v lokálnom časovom pásme - MySQL vždy konvertuje tento čas do UTC.[15]

6.2.4 Amazon Web Services

Amazon Web Services[2] (ďalej iba ako AWS), poskytuje webové služby, takzvané *on-demand clound computing*. Je to typ služby, kedy sú výpočtové zdroje pridelené na základe potreby. V rámci webovej aplikácie bolo zvolené práve AWS pre umiestnenie databázy a súborov (ako sú profilové fotky užívateľov, fotky psov, súborov s výsledkami alebo propozíciami pretekov a tak ďalej). Hlavnou výhodou je ročná bezplatná služba a dostačujúce veľké úložisko pre súbory a databázu tejto webovej aplikácie.[10]

Amazon Simple Storage Service

Amazon Simple Storage Service[3] (ďalej iba Amazon S3) poskytuje úložisko pre zálohovanie dát, zhromažďovanie a analýzu súborov. Poskytuje jednoduchú užívateľskú podporu pre organizovanie a správu dát. Funguje na princípe *buckets*, teda takzvaných vedier, kde konkrétne vedro skladuje dáta pre konkrétnu aplikáciu.[10] Amazon S3 Bucket poskytuje úložisko pre neobmedzený počet objektov (dokumentov, obrázkov a tak ďalej), na prvý rok do veľkosti 5 GB zadarmo. Pre potreby aplikácie a vzhľadom na to, že v rámci aplikácie nie sú uchovávané žiadne citlivé súbory, je prístup k vedru verejný.

Amazon Relational Database Service

Ďalšou službou AWS, ktorá je použitá v rámci webovej aplikácie, je Amazon Relational Database Service[4]. Poskytuje podporu pre viacero databázových systémov, ako je napr. MySQL, PostgreSQL, Amazon Aurora, MariaDB a iné.[10] Používateľom poskytuje jednoduchý nástroj pre vytvorenie a správu databázy, automatické alebo manuálne zálohovanie databázy, automatickú detekciu porúch a zotavenie z nich. V rámci prvého roku zadarmo poskytuje 750 hodín používania mesačne a 20 GB veľké úložisko pre dáta a takisto 20 GB úložisko pre zálohy databázy.

6.2.5 Heroku

Heroku[5] je cloudová platforma PaaS (Platform as a Service), ktorá podporuje niekoľko programovacích jazykov ako napríklad Python, PHP, Java, Node.js, Ruby a iné. Webové aplikácie, ktoré fungujú na Heroku serveri, majú unikátnu doménu, ktorá je používaná k smerovaniu HTTP požiadaviek k správnym kontajnerom, takzvaným dynom. Aplikácia je umiestnená na Heroku server pomocou služby Git.[29] V rámci práce je použitá Free verzia,

[2] https://aws.amazon.com/
[3] https://aws.amazon.com/s3/
[4] https://aws.amazon.com/rds/
[5] https://www.heroku.com/

ktorá poskytuje dostatočné možnosti, aj napriek pomalšej rýchlosti načítavania aplikácie. Aplikáciu je možné nájsť na adrese www.agility-bc.herokuapp.com/.

6.3 Front-end

Front-end, alebo klientska časť, je interaktívna časť aplikácie prístupná užívateľom. Táto časť webovej aplikácie je implementovaná v rámci Djangovho jazyku šablón, ktorý využíva HTML a CSS komponenty.

6.3.1 Bootstrap

Bootstrap[6] je bezplatný, voľne šíriteľný a jeden z najpopulárnejších frameworkov, ktorý kombinuje HTML, CSS a JavaScript prvky. Poskytuje návrhové šablóny pre formuláre, tlačidla, navigačný panel, modálne okná a rôzne iné komonenty webovej stránky. Existuje mnoho šablón, ktoré spájajú tieto jednotlivé komponenty a vytvárajú základné užívateľské rozhranie v jednom štýle. V rámci webovej aplikácie pre organizátorov agility bola použitá jedna z voľne dostupných tém Bootstwatch[7], konkrétne téma Flatly.

6.4 Ďalšie použité technológie

Okrem technológii použitých na strane serveru a na klientskej strane, boli použité aj technológie pre verzovanie a zálohovanie zdrojového kódu programu Git a jeho webová služba Github a webová aplikácia pre lepšie manažovanie času a úloh v rámci vývoja informačného systému (Trello). Tieto technológie sú stručne opísané v nasledujúcich podkapitolách.

6.4.1 Github

Git je bezplatný, voľne šíriteľný verzovací systém. V rámci vývoja webovej aplikácie bola použitá jeho webová služba Github[8]. Poskytuje možnosti zálohovania a verzovania projektu. Okrem iného bol využitý aj na zálohovanie zdrojového kódu pre webovú aplikáciu - odkiaľ zdrojový kód čerpalo `Heroku` pre načítanie a beh aplikácie.

6.4.2 Trello

Pre lepší prehľad nad rôznymi podúlohami jednotlivých častí webovej aplikácie bola použitá aplikácia pre plánovanie - Trello[9]. V rámci tejto aplikácie sa dajú vytvárať nástenky a vrámci nich rôzne stĺpce. Aplikácia bola používaný najmä na sledovanie jednotlivých úloh patriacich do konkrétnej časti aplikácie - jeden stĺpec predstavoval jednu časť aplikácie, ako napríklad: užívateľ, pes, preteky a tak ďalej.

[6] https://getbootstrap.com/
[7] https://bootswatch.com/
[8] https://github.com/
[9] https://trello.com/

Kapitola 7

Návrh systému

Nasledujúca kapitola sa venuje samotnému návrhu systému ktorý vznikal na základe požiadaviek užívateľov, definovaných v kapitole 5. V rámci kapitoly je opísaná použitá architektúra, dátový návrh pomocou ER diagramu. V poslednej časti tejto kapitoly je opísaný návrh užívateľského rozhrania, ktorý je určený primárne pre desktopové zariadenia - ale aplikácia môže byť využívaná aj na mobilných zariadeniach.

7.1 Architektúra systému

V kapitole 2 boli predstavené najpoužívanejšie typy architektúr webových aplikácií. Vzhľadom na potreby užívateľov, ako je možnosť sa prihlasovať na dané preteky z akéhokoľvek zariadenia a na samotnú nenáročnosť na dané zariadenia užívateľa, bola zvolená **trojvrstvová architektúra**.

Trojvrstvová architektúra bola zvolená najmä pre jej možnosť oddeliť jednotlivé vrstvy aplikácie. Databázový server (dátová vrstva) funguje prostredníctvom Amazon Web Services - Relational Database Service centrálne, pre všetkých užívateľov - uchováva si v rámci jednej databázy informácie o všetkých užívateľoch, psoch, vytvorených tímoch, pretekoch a tak ďalej. Samotná aplikácia je umiestnená na serveri Heroku, kde je zdrojový kód importovaný z verzovacieho systému Github - pri nahratí novej verzie na Github, je táto verzia hneď preložená a importovaná na Heroku - ktoré predstavuje aplikačnú vrstvu. Samotná prezentačná vrstva je koncové zariadenie užívateľa - webový prehliadač.

Samotná aplikácia je implementovaná v Djangu a drží sa architektonického vzoru **Model-View-Template**. Princíp tohto architektonického vzoru je priblížený v kapitole 6.

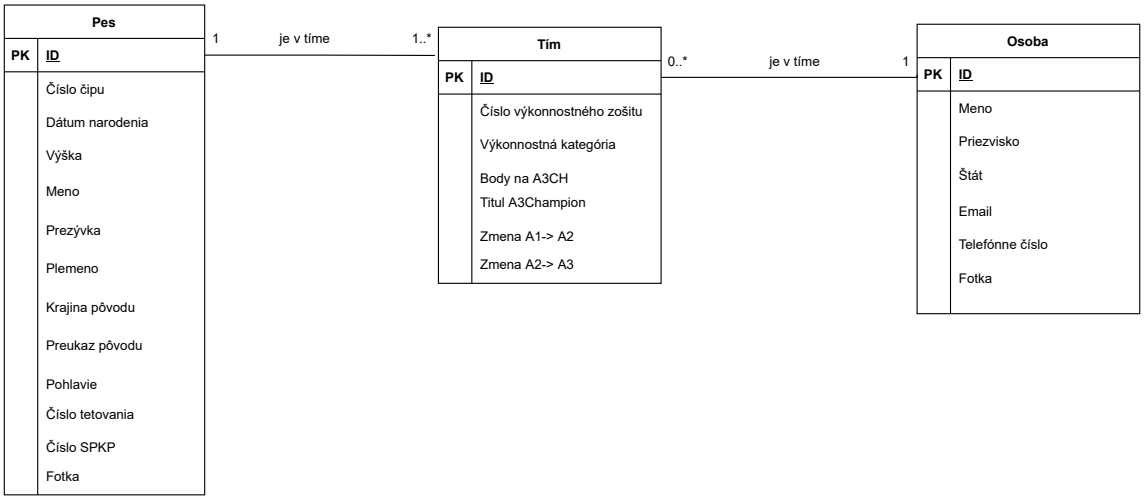
7.2 ER diagram

Entitno-vzťahový diagram, ďalej iba ER diagram (z angl. entity-relationship diagram) je konceptuálny model, ktorý sa používa pre návrh a znázornenie objektov - **entít** a **vzťahov** medzi nimi v systéme. Entity, takisto ako vzťahy, môžu mať rôzne vlastnosti - **atribúty**. Medzi základné prvky okrem už spomenutých prvkov patrí aj **entitná množina** - množina entít rovnakého typu, ktorá má rovnaké atribúty a **vzťahová množina** - množina vzťahov rovnakého typu.[16, sl. 15]

V nasledujúcich sekciách budú postupne priblížené jednotlivé entitné množiny a vzťahy medzi nimi. Celkový ER diagram sa následne nachádza v prílohe B.

Pes - Tím - Osoba

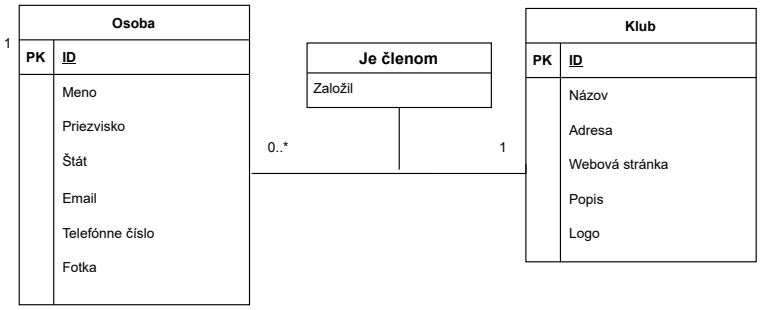
Hlavnými entitnými množinami nachádzajúcimi sa v informačnom systéme sú pes a osoba. Entitná množina Osoba môže predstavovať viacero užívateľov webovej aplikácie - či už samotného pretekára alebo organizátora pretekov. Oba typy osôb môžu v rámci informačného systému vykonávať viacero funkcií - a jedna a tá istá osoba môže organizovať preteky a zároveň sa ich aj zúčastňovať. Pretekov sa ale nezúčastňuje samotná osoba/pes, ale Tím, ktorý spolu tvoria. V rámci jedného tímu môže byť iba jeden pes a jedna osoba - pretekár. Jeden pes/pretekár môžu byť ale členmi viacerých tímov - preto bol zvolený vzťah 1 ku 1..*. Tím začína vo výkonnostnej kategórii A1, následne sa môže posúvať do vyšších kategórií (A2/A3). Dátum a čas tejto zmeny kategórie je v systéme uchovávaný.



Obr. 7.1: Vzťah medzi psom, osobou a tímom, ktorí spolu tvoria

Osoba - Klub

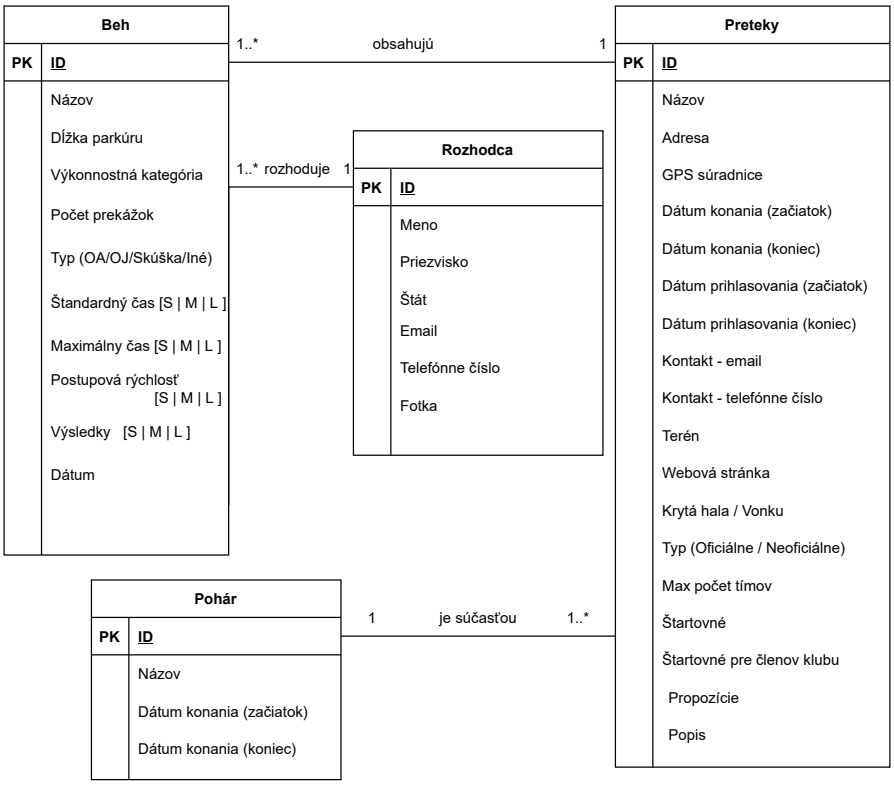
Kluby sú organizácie, ktoré združujú pretekárov. Jedna osoba môže byť členom iba jedného klubu, počet členov klubu ale nie je vôbec obmedzený (vzťah 1 ku 0..*), člen klubu môže byť zároveň jeho zakladateľom (atribút `založil`), z ktorej funkcie mu plynie viacero oprávnení ako je napríklad úprava informácií o klube. Klub môže mať viacero zakladajúcich členov.



Obr. 7.2: Vzťah medzi osobou a klubom

Pohár - Preteky - Beh - Rozhodca

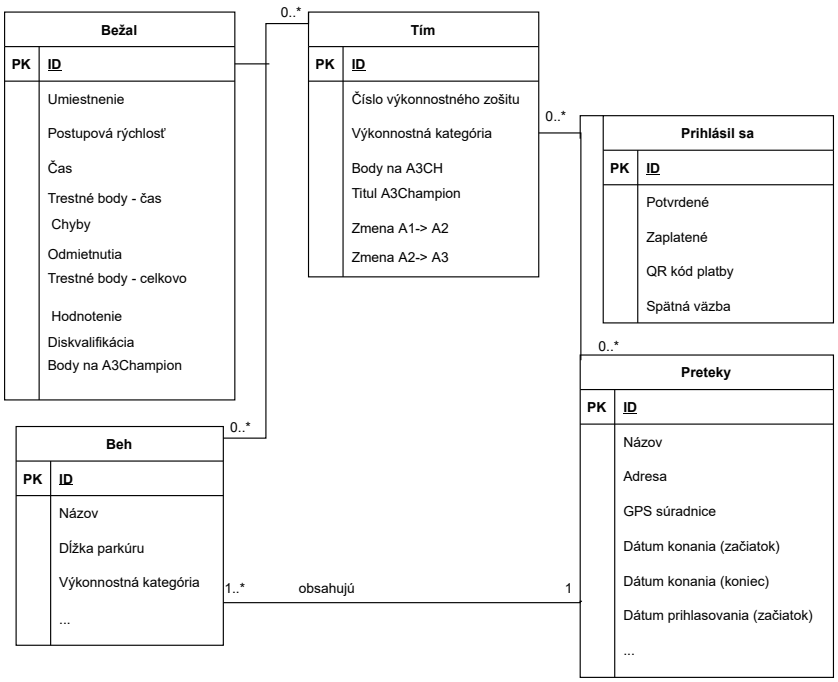
Každý beh rozhoduje práve jeden rozhodca, ale rozhodca môže rozhodovať viacero behov (vzťah 1 ku *). Rozhodca ale v rámci informačného systému nemá žiadne funkcie a teda je modelovaný zvlášť od entitnej množiny Osoba. Každá výšková kategória psov behá rovnaký parkúr, s rovnakou dĺžkou a prekážkami - môže sa líšiť v postupovej rýchlosti, maximálnom a štandardnom čase. Z tohto dôvodu sú tieto atribúty uchovávané zvlášť pre každú výškovú kategóriu. Každý beh musí byť súčasťou nejakých pretekov - v rámci jedných pretekov môže byť niekoľko rozdielnych behov, ale beh sa môže konať iba ako súčasť jedných pretekov (vzťah 1 ku 1..*). Viaceré preteky môžu ale nemusia byť súčasťou pohára (vzťah 1 ku 1..*).



Obr. 7.3: Vzťah medzi pretekom, behmi v rámci neho, pohárom a rozhodcom

Tím - Preteky - Beh

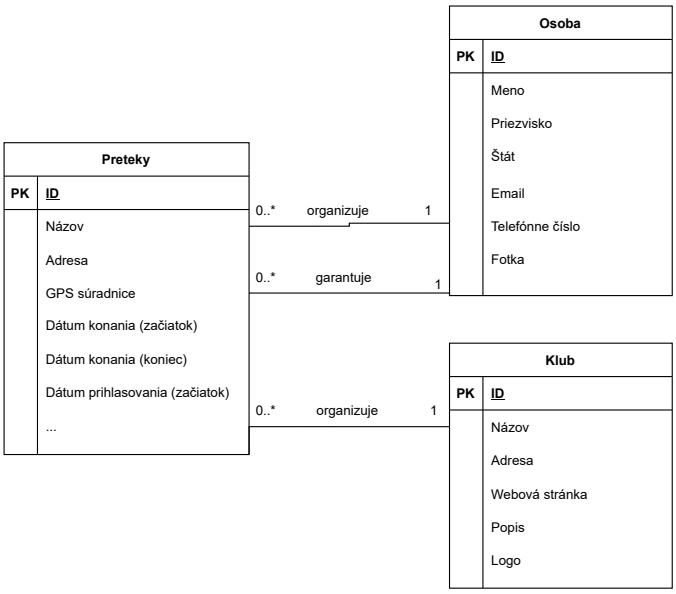
Tím sa môže prihlásiť na viacero pretekov, a takisto v rámci jedných pretekov môže byť prihlásených viacero tímov (vzťah 0..* ku 0..*) - tento vzťah má aj viacero atribútov vzhľadom na stav prihlášky. Druhým, rovnako dôležitým vzťahom, je vzťah medzi entitnými množinami Tím a Beh - *Bežal*. V rámci tejto vzťahovej množiny sú v databáze uložené údaje o výsledku daného tímu, ktorý bežal už konkrétny beh. Jeden tím mohol bežať viacero behov, takisto jeden beh mohlo bežať viacero tímov - vzťah 0..* ku 0..*.



Obr. 7.4: Vzťah medzi pretekmi, behom a tímom, ktorý sa daných pretekov zúčastňuje

Preteky - Osoba - Klub

Poslednou súčasťou ER diagramu je organizácia pretekov - preteky môže organizovať fyzická osoba, prípadne klub. Pre každé preteky sa určuje práve jedna fyzická osoba, ktorá dané preteky garantuje - jedna osoba môže garantovať viacero pretekov (vzťah 0..* ku 1). Jedna osoba/klub môže organizovať viacero pretekov, pričom preteky majú práve jedného organizátora (vzťah 0..* ku 1).

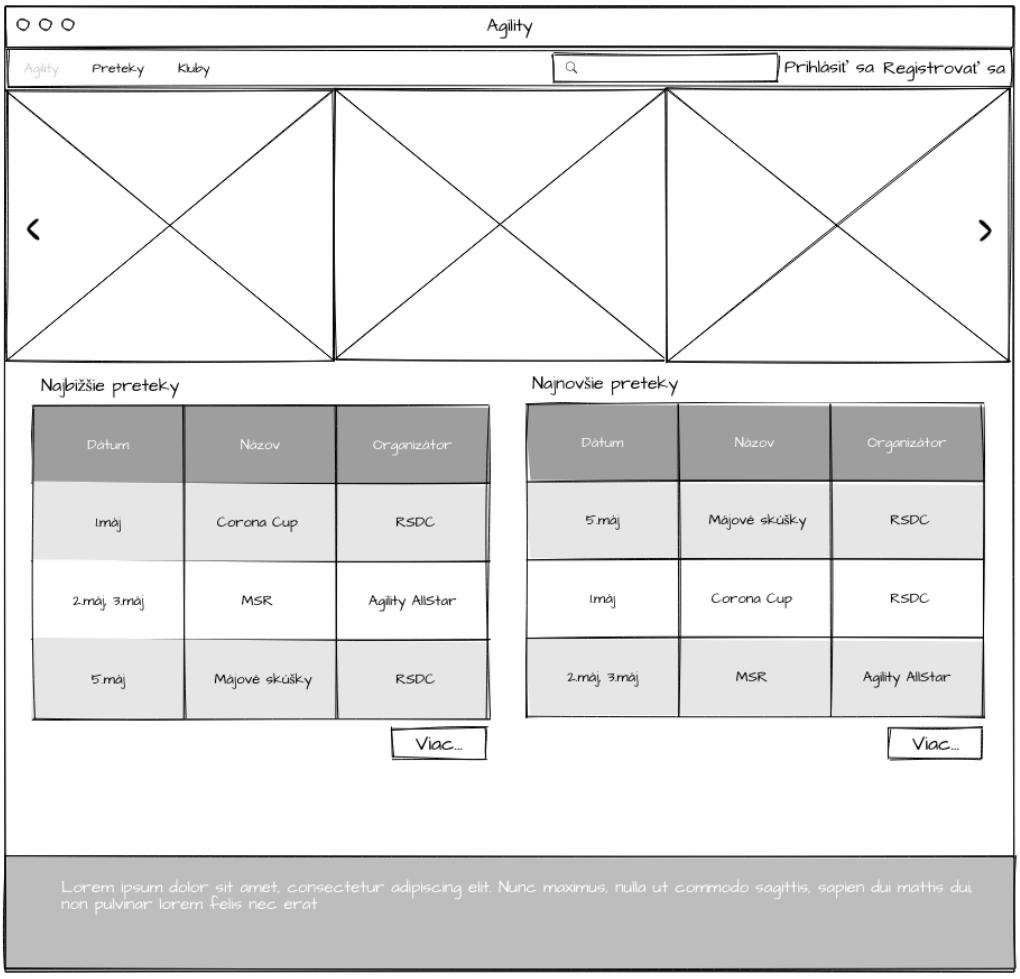


Obr. 7.5: Vzťah medzi pretekmi, osobou a klubom

7.3 Návrh užívateľského rozhrania

Poslednou časťou návrhu aplikácie bol návrh užívateľského rozhrania. Hlavným cieľom vyvíjaného užívateľského rozhrania bola prehľadnosť a jednoduchosť. Výsledný návrh bol z tohto dôvodu konzultovaný s pretekármi a organizátormi agility, pre ktorých je daná aplikácia vyvíjaná.

Na obrázku 7.6 sa nacháza návrh úvodnej stránky, ktorá zobrazuje užívateľovi všetky potrebné informácie. V hornej časti stránky sa nachádza navigačný panel, prostredníctvom ktorého má užívateľ možnosť sa prekliknúť na preteky a zobraziť si kluby. V pravej časti navigačného panela sa nachádza vyhľadávanie a ak užívateľ nie je prihlásený, má možnosť sa prihlásiť, prípadne sa registrovať. Ak užívateľ má vytvorený účet a je prihlásený, vidí menu, prostredníctvom ktorého má možnosť sa prekliknúť na svoj profil, prípadne pridať nového psa alebo preteky. V hlavnom tele stránky vidí užívateľ banner, pomocou ktorého sa tiež vie prekliknúť k pretekom. Pod týmto bannerom sa nachádzajú dve tabuľky - jedna tabuľka, ktorá zobrazuje nadchádzajú preteky a druhá tabuľka, ktorá zobrazuje najnovšie pridané preteky.



Obr. 7.6: Návrh hlavnej stránky

Kapitola 8

Implementácia

V nasledujúcej kapitole bude opísaná implementácia webovej aplikácie tak, aby spĺňala požiadavky organizátorov pretekov Agility aj samotných pretekárov. Najskôr bude opísaná implementácia serverovej časti, pre ktorú bol zvolený programovací jazyk Python 3 spolu s jeho frameworkom Django. Následne bude opísaná klientská časť, ktorá je vytvorená pomocou šablónovacieho jazyka poskytnutého frameworkom Django spolu v kombinácii s HTML značkovacím jazykom, kaskádovými štýlmi (CSS) a skriptovacím jazykom JavaScript a jeho knižnicou JQuery.

8.1 Štruktúra aplikácie

Aplikácia pozostáva z viacerých menších častí, ktoré spolu vzájomne komunikujú a spolupracujú. Každá menšia časť - aplikácia má na starosti jednu časť výslednej aplikácie. Okrem priečinkov, ktoré predstavujú jednotlivé čiastkové aplikácie, je ešte hlavný priečinok, ktorý spúšťa celú aplikáciu. Adresárová štruktúra aplikácie vyzerá následovne:

- **/agility_main/** - aplikácia, ktorá má na starosti základnú funkcionalitu ako je vyhľadávanie, kontaktný formulár a hlavnú stránku. V rámci adresára `templates` okrem šablón pre vyššie spomenuté stránky, obsahuje tiež aj základnú šablónu, ktorú rozširujú jednotlivé šablóny aplikácií.
- **/bc_project/** - hlavný priečinok, ktorý vznikol pri vytvorení Django projektu (aplikácie).
  - **/bc_project/___init___.py** - prázdny súbor, slúži na označenie priečinku ako `balíček` (angl. `package`).
  - **/bc_project/asgi.py**
  - **/bc_project/settings.py** - hlavný súbor celej aplikácie. Slúži na zahrnutie všetkých aplikácií, nastavení pre databázu, pre úložisko súborov a všetky ostatné nastavenia potrebné pre bezproblémové fungovanie aplikácie.
  - **/bc_project/urls.py** - tento súbor obsahuje všetky URL adresy aplikácie (prípadne zahrňuje všetky URL všetkých aplikácií).
  - **/bc_project/wsgi.py** - spolu so súborom **asgi.py** obsahujú všetky potrebné informácie a nastavenia pre funkčnosť aplikácie na serveroch, ako je napríklad Apache a iné.

- **/clubs/** - táto aplikácia zahŕňa logiku ohľadne klubov - vytvorenie nového klubu, upravenie existujúceho, zobrazenie klubu a iné funkcie.
- **/competitions/** - aplikácia obsahuje funkcionalitu pre vytvorenie a úpravu pretekov, prihlásenie tímov, kontakovanie organizátora/pretekárov a mnoho iného.
- **/cups/** - aplikácia zahŕňa jednoduchú logiku pre zaradenie aplikácie do pohára, zobrazenie pretekov v rámci jedného pohára a iné.
- **/dog/** - jedna z najdôležitejších aplikácii vyvíjaného systému. Tento balíček obsahuje funkcie pre pridanie nového psa, vytvorenie tímu, zobrazenie profilu psa spolu s jeho výsledkami a tak ďalej.
- **/judge/** - táto aplikácia zahrňuje funkcionalitu ohľadne rozhodcov - pridanie nového rozhodcu, upravenie už existujúceho rozhodcu, zobrazenie rozhodcu spolu s pretekmi, ktoré rozhodoval/bude rozhodovať a tak ďalej.
- **/run/** - v rámci jedného preteku môže byť hneď niekoľko behov - a presne pre tieto behy slúži aplikácia `run`. V tomto balíčku sa nachádza logika za pridaním nového behu - spolu s nahratím alebo manuálnym zadaním výsledkov jednotlivých tímov.
- **/userAccount/** - v rámci tejto aplikácie je definovaná všetka aplikačná logika, ktorá implementuje užívateľa - registrácia a prihlásenie, zobrazenie profilu (spolu so všetkými jeho funkciami), obnovenie hesla a tak ďalej.
- **manage.py** - súbor, ktorý slúži pre spustenie aplikácie z príkazového riadku v rámci localhostu pomocou príkazu `manage.py runserver`. Jeho prostredníctvom sa takisto nahrávajú zmeny do databázy. Tento súbor bol vygenerovaný automaticky pri vytvorení Django projektu, počas implementácie aplikácie s ním nebolo manipulované.
- **requirements.txt** - súbor, ktorý obsahuje všetky potrebné Python balíčky spolu s ich verziami - tento súbor sa používa aj v rámci umiestnenia aplikácie na server Heroku
- **runtime.txt** - takisto súbor, ktorý je potrebný pre beh aplikácie na serveri Heroku - obsahuje potrebnú verziu programovacieho jazyku Python.

Jednotlivé aplikácie majú rôzne funkcie na základe toho aké funkcie výslednej aplikácie spĺňajú. Každá aplikácia má viac-menej rovnakú štruktúru a to nasledovnú (ukázané na štruktúre balíčku `competitions`):

- **/competitions/templates/competitions** - v rámci každého balíčku sa nachádza adresár `templates`, v ktorom je ďalší adresár, ktorý sa nazýva rovnako ako aplikácia. Toto slúži najmä pre jednoduchosť zdieľania kódu šablóny medzi rôznymi aplikáciami.
- **/competitions/___init___.py** - slúži na označenie adresára ako Python balíček.
- **/competitions/admin.py** - v rámci tohto súboru sú registrované jednotlivé modely (tabuľky) pre Django administrátora - k tomu majú prístup užívatelia s rolou admin.
- **/competitions/apps.py** - obsahuje konfiguráciu aplikácie.
- **/competitions/filters.py** - slúži na definovanie tried, podľa ktorých sú filtrované jednotlivé údaje v tabuľkách modelu.

- **/competitions/forms.py** - obsahuje formuláre, ktoré sú použité v rámci aplikácie. Môže obsahovať formuláre, ktoré obsahujú atribúty podľa modelu (tabuľky, do ktorej sa dané údaje z formulára ukladajú) alebo aj vlastné definované formuláre s vlastnými atribútmi (ktoré nededia z modelu).
- **/competitions/models.py** - v tomto súbore sú ako triedy definované jednotlivé tabuľky databázy - spolu s ich atribútmi. Okrem toho sa môžu v tomto súbore nachádzať rôzne funkcie pre daný model.
- **/competitions/signals.py** - funkcie v tomto súbore slúžia na upravenie/doplnenie údajov pred/po uložení instancie modelu.
- **/competitions/urls.py** - slúži na uloženie URL adries aplikácie - a spúšťa jednotlivé funkcie v súbore views.py (podľa toho, aká URL adresa je načítavaná).
- **/competitions/views.py** - funkcie, ktorých úlohou je načítanie správnej šablóny, prípadne presmerovanie na inú funkciu, spracovanie formulárov a iné funkcie.

8.2 Serverová časť (Back-end)

Ako bolo spomenuté na začiatku tejto kapitoly, pre implementáciu serverovej časti aplikácie bol zvolený Python framework Django a pre uloženie dát bol zvolený databázový systém PostgreSQL. V tejto podkapitole je popísaná implementácia databázy, autentizácia užívateľov, prepojenie s Google účtom, integrácia Google kalendára a Google máp a získavanie asociačných pravidiel pre analýzu úspešných plemien.

8.2.1 Databáza

Od prvotného návrhu systému bola schéma databázy niekoľkokrát upravovaná, vzhľadom na uvedomenie si nových informácií, ktoré sú potrebné, aby boli uchovávané v rámci aplikácie. Ako bolo spomenuté v kapitole 6, databáza je umiestnená na serveri AWS a samotná aplikácia beží na serveri Heroku. Z tohto dôvodu bolo nutné definovať v súbore `settings.py` potrebnú konfiguráciu, pre pripojenie sa na databázu.

Definícia jednotlivých tabuliek databázy sa nachádza v súboroch `models.py`. Každá tabuľka je definovaná ako trieda. Každá trieda obsahuje implicitne definovaný atribút - takzvaný primárny kľúč tabuľky. Primárny kľúč je celé číslo, ktoré sa každou vzniknutou instanciou modelu inkrementuje o 1. Tento atribút je unikátny v rámci triedy. Na príklade zdrojového kódu 8.1 je ukázaná definícia tabuľky Tím. Je tu možné vidieť, ako prebieha definícia cudzieho kľúču (angl. foreign key), ktorým sa odkazujeme na objekt uložený v odkazovanej tabuľke - prakticky to znamená uloženie si primárneho kľúča daného objektu z odkazovanej tabuľky ako kľúču cudzieho. Pri definícii tohto cudzieho kľúča nesmieme zabudnúť definovať, čo sa stane, ak je odkazovaný objekt vymazaný. V tomto konkrétnom príklade sa nastaví hodnota cudzieho kľúču na `NULL` - už neodkazujeme na konkrétneho pretekára alebo psa, ale stále si v databáze uchovávame informáciu o existencii tohto tímu (najmä z dôvodu uchovania výsledkov). Nie vždy musí byť zvolená táto možnosť - môže byť zvolená možnosť, kedy sa vymažú všetky záznamy závislej entity (príkaz `models.CASCADE)`.

```
1  from django.db import models
2  from django.core.validators import MaxValueValidator
3
4  from userAccount.models import Account
5  from competitions.models import Competition
6
7  class Team(models.Model):
8      competitor = models.ForeignKey(Account, null = True, on_delete = models
           .SET_NULL)
9      dog = models.ForeignKey(Dog, null=True, on_delete=models.SET_NULL)
10
11     numberWB = models.CharField(max_length=15, unique=True)
12     performanceCategory = models.CharField(max_length=7, choices=
           PERFORMANCE_CATEGORY, default='A1')
13     pointsA3CH = models.PositiveIntegerField(default=0, validators=[
           MaxValueValidator(100)]
14     changeA1A2 = models.DateTimeField(blank=True, null=True)
15     changeA2A3 = models.DateTimeField(blank=True, null=True)
```

Výpis 8.1: Implementácia tabuľky Tím

8.2.2 Užívateľ a autentizácia užívateľa

Pre uchovávanie si informácií o pretekároch a užívateľov bola vytvorená vlastná trieda `Account`. Vlastná trieda pre užívateľa bola zvolená najmä z dôvodu uchovania viacerých informácií o užívateľovi, ktorá trieda `User` neposkytuje. Takisto výhodou definovania vlastnej triedy bola možnosť definovať e-mail ako atribút potrebný pre prihlásenie a tým sa uistiť, že tento atribút bude v rámci databázy unikátny.

Užívatelia sa prihlasujú pomocou e-mailovej adresy a hesla. Pre autentizáciu užívateľa poskytuje Django základné funkcie v balíčku `django.contrib.auth`. Okrem možnosti vytvorenia klasického účtu, majú užívatelia možnosť sa prihlásiť pomocou Google účtu (prípadne prepojiť už existujúci účet s Google účtom), čo je implementované pomocou knižnice `django-allauth`, ktorá umožňuje prihlásenie, registráciu a autentizáciu užívateľov prostredníctvom účtov aplikácií tretích strán. Google poskytovateľ používa protokol OAuth2, vďaka ktorému užívateľ nemusí zdielať svoje prihlasovacie údaje do aplikácie Google s vyvíjanou webovou aplikáciou. Pre umožnenie prepojenia účtu s Google účtom, bolo potrebné vytvoriť projekt v rámci `Google Developers` a následne uložiť konfiguračné informácie - ID klienta a tajný kľúč klienta, do tabuľky `Social application` a definovať potrebné údaje v súbore `settings.py`.

Iba užívateľ, ktorý má rolu admin, má právo upravovať informácie o akomkoľvek preteku, klube a tak ďalej. Ostatní užívatelia môžu upravovať informácie o pretekoch iba v prípade ak sú organizátorom alebo garantom, prípadne zakladateľom klubu, ktorý preteky organizuje. Tieto oprávnenia sú definované pomocou niekoľkých dekorátorov, ktoré sú definované v súbore `agility_main/decorators.py`. Na ukážke zdrojového kódu 8.2, je možné vidieť definíciu dekorátora, ktorý kontroluje, či prihlásený užívateľ je zakladateľom klubu alebo prípadne má rolu `is_admin`, ktorá značí, že je správcom celého systému.

```python
from django.shortcuts import redirect
from django.contrib import messages

def founder_club(view_function):
    def wrapper(request, *args, **kwargs):
        if request.user.founderClub or request.user.is_admin:
            return view_function(request, *args, **kwargs)
        else:
            messages.warning(request, 'Uzivatel nema dostatocne opravnenia
                upravovat informacie o~klube.')
            return redirect('profileClub', *args, **kwargs)
    return wrapper
```

Výpis 8.2: Implementácia dekorátora @founder_club

8.2.3 Pohľady

Pohľady sú funkcie, ktoré prijímajú HTTP požiadavky a vracajú HTTP odpovede. Všetky pohľady sú definované v príslušných súboroch `views.py` v adresároch jednotlivých aplikácií. Django umožňuje programátorom dva spôsoby definovania pohľadov - pomocou funkcií alebo pomocou tried. Vzhľadom na vyvíjanú aplikáciu a samotnú zložitosť niektorých úloh, ako je napríklad spracovanie dvoch formulárov z jednej stránky, boli zvolené pohľady definované pomocou funkcií. Pre konzistenciu kódu sú všetky pohľady definované rovnakým spôsobom.

Na ukážke kódu 8.3 je možné vidieť definíciu pohľadu, ktorý je zavolaný po zadaní URL adresy aplikácie. Je vidieť, že funkcia prijíma HTTP požiadavku, následne definuje potrebné informácie, ktoré sú zobrazované na hlavnej stránke užívateľovi a odovzdáva ich šablóne pomocou slovníka `context`.

Na podobnom princípe, ako ukážka kódu 8.3 sú definované všetky pohľady tvorené v rámci aplikácie. V prípade spracovania formulára sa dáta vyplnené vo formulári získavajú z HTTP požiadavky poslanej v premennej `request` zadaním konkrétnej metódy, s ktorou bol daný formulár definovaný, napríklad `request.POST`. Nahrané obrázky, prípadne rôzne dokumenty, sú získavané z formulára pomocou príkazu `request.FILES`.

```python
from django.shortcuts import render, redirect
from competitions.models import Competition

def home(request):
    competitions_new = Competition.objects.filter(startDate__gte=date.today
        ()).order_by('startDate')[:5]
    competitions_created = Competition.objects.filter(createdDate__lte=date
        .today()).order_by('createdDate')[:5]
    context = {
        'competitions': competitions_new,
        'competitions_created': competitions_created
    }
    return render(request, 'homepage/homepage.html', context)
```

Výpis 8.3: Implementácia domovskej stránky

Mapovanie URL adries na pohľady

Ako bolo ukázané na obrázku 6.3, mapovanie URL adries na pohľady sa nachádza v súboroch `urls.py`. Na príklade 8.4 je vidieť mapovanie URL adries aplikácie, ktorá spracováva domovskú stránku, vyhľadávanie a kontaktný formulár. Mapovanie jednotlivých URL adries vyzerá nasledovne: najskôr, je zadefinovaná samotná URL adresa, nasleduje argument, ktorým sa volá konkrétna funkcia pohľadu a následne voliteľný argument, ktorý pomenováva konkrétnu URL adresu.

Aby hlavná aplikácia spolupracovala so všetkými ostatnými čiastkovými aplikáciami, je potrebné v hlavnej aplikácii, v adresári `bc_project`, zahrnúť v súbore `urls.py` všetky súbory s definovanými URL adresami. Definícia prebieha podobne ako je ukázané na príklade 8.4, len s tým rozdielom, že nie sú volané konkrétne pohľady, ale sú zahrnuté súbory, ktoré definujú tieto URL adresy, napríklad: `path('', include('agility_main.urls'))`.

```
1 from django.urls import path
2 from agility_main import views
3
4 urlpatterns = [
5     path('', views.home, name='home'),
6     path('search/', views.search, name='search'),
7     path('contact/', views.contact, name='contact'),
8 ]
```

Výpis 8.4: Mapovanie konkrétnych URL adries

8.2.4 Google API

Okrem prepojenia užívateľského účtu majú užívatelia možnosť vidieť preteky aj pomocou Google kalendára a zobraziť si mapu nadchádzajúcich pretekov, prípadne mapu miesta konania sa konkrétnych pretekov. Obe tieto funkcie sú implementované pomocou `Google API`[1].

U Google kalendára, implementovaného pomocou `Google Calendar API`[2], má užívateľ možnosť zobraziť si na svojom profile Google kalendár so všetkými nadchádzajúcimi pretekmi. Pre možnosť pracovať s Google kalendárom bolo nutné povoliť autorizáciu pomocou `OAuth 2.0`.

Pre implementáciu Google máp pre zobrazenie pretekov alebo miesta konania pretekov, bolo potrebné použiť `Maps JavaScript API`[3] a `Places Library`[4]. Pri inicializácií Google mapy bolo nutné určiť samotné priblíženie mapy a daný stred, ktorý ale nebol až tak dôležitý, keďže po zadaní a vyhľadaní adresy konania pretekov, je tento stred premiestnený na bod zemepisnej dĺžky a šírky danej adresy. Na tomto mieste je vytvorený takzvaný špendlík (angl. pin), ktorý ho označuje.

8.2.5 Asociačné pravidlá

Pre analýzu najúspešnejších plemien bol použitý algoritmus Apriori vysvetlený v kapitole 3. Pre detailnejšiu analýzu bola vytvorená tabuľka úspešných psov `SuccessfullBreeds`, ktorá

[1] API - Application Programming Interface
[2] https://developers.google.com/calendar/overview
[3] https://developers.google.com/maps/documentation/javascript/overview
[4] https://developers.google.com/maps/documentation/javascript/places

obsahuje plemeno psa, jeho výškovú kategóriu, pohlavie, či má tento pes papiere pôvodu a výkonnostnú kategóriu. Do tejto tabuľky sú nahrávané úspešné psy automaticky, po nahratí výsledkov daného behu. Následne bola táto tabuľka spracovaná a použitá funkcia `apriori` z knižnice `apyori` pre vygenerovanie asociačných pravidiel.

8.2.6 Platobná brána

Pri implementácií platobnej brány bol zistený problém, ktorý nebol známy pri samotnej analýze požiadavkov, a to konkrétne fakt, že jednotlivé platby za preteky môžu ísť na predom neznámy počet účtov organizátorov. Po dôslednej analýze možných riešení a snažení sa nájsť riešenia, kedy by bolo možné využiť platobnú bránu pre platbu za účasť na pretekoch, bola samotná platobná brána neimplementovaná. Namiesto toho bola zvolená možnosť automatického generovania QR kódov pre danú platbu pomocou služby Pay by square[5], ktorú využíva knižnica `pay_by_square` pre vytvorenie kódu platby, ktorý je neskôr použitý pre vygenerovanie QR kódu (obrázku) pomocou funkcií z knižnice `qrcode`.

8.3 Klientská časť (Front-end)

V nasledujúcej podkapitole je opísaná implementácia užívateľského rozhrania, ktoré bolo implementované pomocou jazyka šablón, ktorý poskytuje framework Django a spolu s ním bola použitá knižnica `Chart.js`, pre zobrazenie animovaných a interaktívnych grafov výkonnosti psa.

8.3.1 Šablóny

Jazyk šablón, poskytnutý frameworkom Django, bol použitý pre generovanie dynamických webových stránok. Na ukážke kódu 8.5 je možné vidieť šablónu, ktorá implementuje kontaktný formulár pre užívateľov webovej aplikácie. Táto šablóna dedí zo spoločného základu stránok, ktorý sa nachádza v súbore `base/base.html`. Django jazyk šablón umožňuje rozširovať túto šablónu v presne definovaných miestach pomocou kľúčového slova `{% block content %}` rozširuje základnú šablónu v mieste definovanom týmto blokom. Blokov môže byť definovaných viacero, napríklad `content` označuje blok, ktorý sa nachádza medzi značkami `<body> </body>`. V rámci jednej šablóny je možné zahrnúť aj kód z iných šablón, a takto využiť rovnaký kód vo viacerých šablónach bez nutnosti písania rovnakého kódu vo viacerých šablónach.

```
1 {% extends 'base/base.html' %}
2
3 {% block content %}
4     <div class="jumbotron">
5         {% include 'base/messages.html' %}
6         <h4> Mate nejake otazky a nenasli ste odpoved? Kontaktuje nas. </h4>
7         {% include 'base/contact_form.html' %}
8     </div>
9 {% endblock %}
```

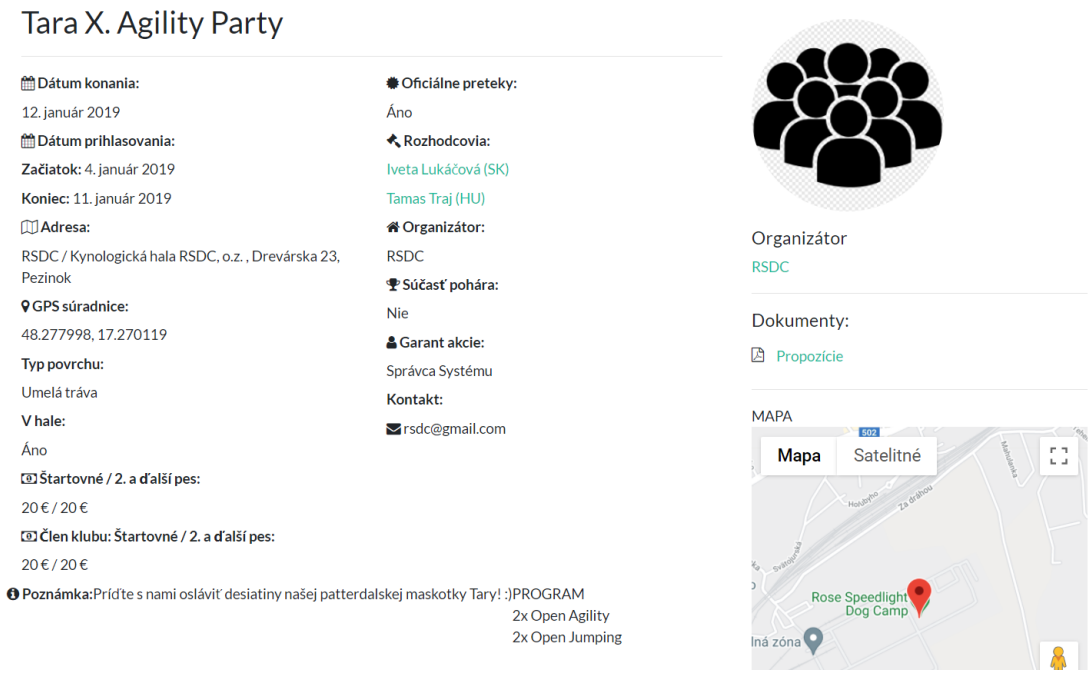
Výpis 8.5: Šablóna pre kontaktný formulár

[5] https://bysquare.com/pay-by-square/

8.3.2 Grafické užívateľské rozhranie

Pri tvorbe grafického užívateľského rozhrania bol kladený dôraz najmä na jednoduchosť a prehľadnosť. Cieľom bolo, aby užívateľ pri hľadaní informácií nestrávil dlhý čas, ale všetky informácie videl prehľadne a na miestach, kde ich očakáva.

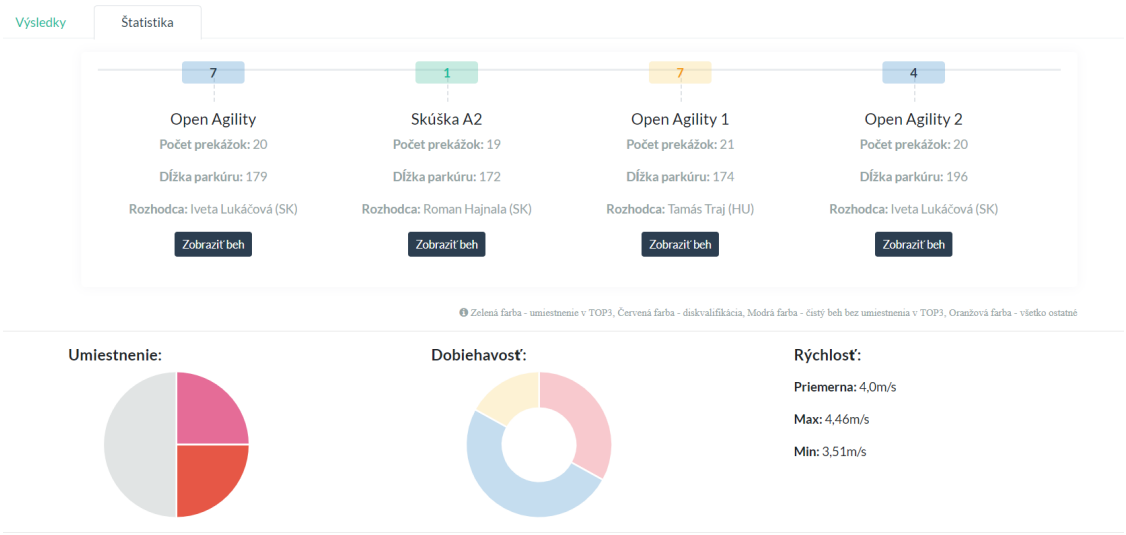
Na obrázku 8.1 je možné vidieť zobrazenie konkrétnych pretekov, spolu so všetkými informáciami, ktoré užívateľ potrebuje hneď na jednom mieste - od dátumu konania, adresy a GPS súradníc, štartovného, organizátora a rozhodcov, až po zobrazenie mapky s miestom konania. Nižšie vidí užívateľ ďalšie informácie o pretekoch, ako aj rôznu funkcionalitu - registrácia na preteky, kontaktovanie organizátora a iné.



Obr. 8.1: Zobrazenie konkrétnych pretekov

Zobrazenie psa

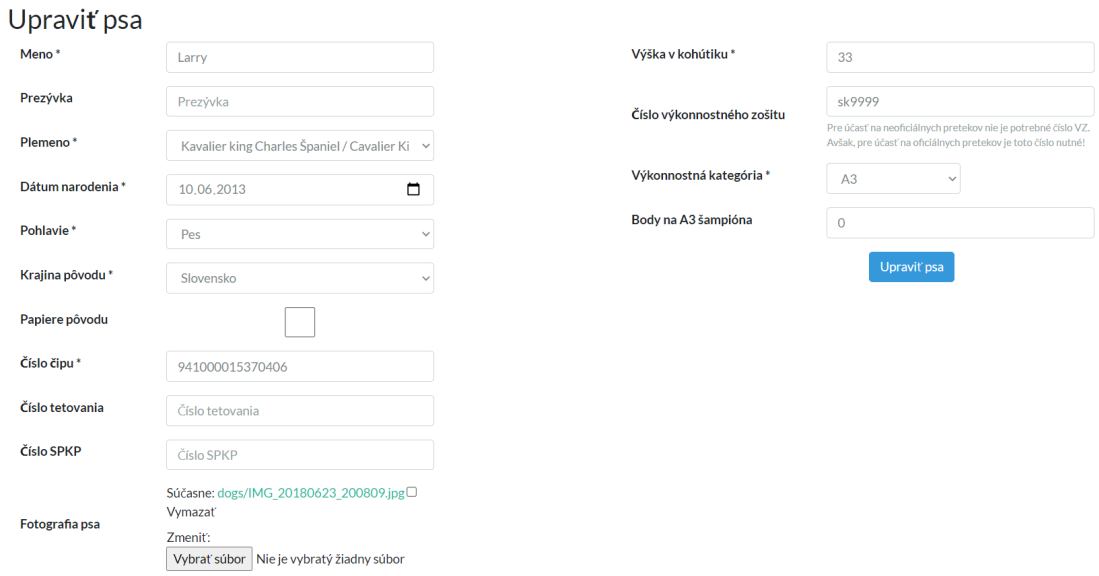
Pri implementácii zobrazenia konkrétneho psa bolo myslené najmä na prehľadnosť a nutnosť mať všetky informácie na jednom mieste. Z toho dôvodu, nie je potrebné navštevovať viacero stránok a na jednej stránke sa nachádzajú výsledky psa spolu aj s prehľadnou štatistikou, oddelené každé na príslušnej karte profilu. Táto štatistika bola implementovaná pomocou javascriptovej knižnice `Chart.js`, konkrétne boli použité jej `Pie chart` a `Doughnut chart`. Oba grafy sú vkladané do šablóny pomocou párovej značky `<canvas>` , napríklad `<canvas id="pie-chart"></canvas>` značí umiestnenie koláčového grafu, ktorý zobrazuje štatistiku dobiehavosti psa. Informácie pre daný graf sú vždy definované v javascriptovom kóde, pomocou premennej `config`. Následne je tento graf vytvorený pomocou príkazu `window.myPie = new Chart(ctxPie, configPie);`, kde premenná `ctxPie` označuje HTML element, kde sa má daný graf vykresliť. Výsledné grafy je možné vidieť na obrázku 8.2



Obr. 8.2: Zobrazenie štatistiky psa

Upravenie psa a tímu

Súčasťou vyvíjaného informačného systému boli aj formuláre. Na obrázku 8.3 je možné vidieť formulár, pre zmenu informácií o psovi. Formulár je dopredu pred-vyplnený s aktuálnymi informáciami o psovi a tíme. V ľavej časti sa nachádzajú základné informácie o psovi ako je jeho meno, plemeno, dátum narodenia a iné. V pravej časti sa zas nachádzajú informácie ohľadne vytvoreného tímu (pes a prihlásený užívateľ), ako je výkonnostný zošit, výkonnostná kategória a prípadne body na A3 šampióna. Ak sa pes nenachádza vo výkonnostnej kategórii A3, užívateľ nemôže vyplniť pole, ktoré obsahuje body na A3 šampióna. Takisto systém upozorní užívateľa, ak už splnil všetky potrebné podmienky na možnosť prestupu do vyššej výkonnostnej kategórie.



Obr. 8.3: Upravenie psa a tímu, ktorý prihlásený užívateľ s ním tvorí

Kapitola 9

Testovanie

Finálnou, ale rovnako dôležitou časťou vývoja aplikácie bolo testovanie, ktoré slúži k overeniu funkčnosti informačného systému pred nasadením do každodenného používania. Testovanie prebiehalo v dvoch fázach: počas implementácie bola aplikácia testovaná programátorom a následne, po dokončení implementácie bola aplikácia testovaná na vzorke potencionálnych používateľov.

9.1 Testovanie programátorom

Výsledná aplikácia má širokú funkcionalitu, ktorú bolo nutné dôkladne otestovať. K testovaniu dochádzalo po implementácii každej novej funkcionality, vďaka čomu boli jednoducho odhalené samotné chyby spojené s danou funkcionalitou, ako je komunikácia so serverom, prípadne chyby spojené s operáciami nad databázou, rôzne chyby v predávaní parametrov do šablóny a mnoho iného.

V poslednej fázy vývoja bola webová aplikácia testovaná ako celok - od úplne základných funkcií ako je vytvorenie účtu (ktorý nemá práva administrátora), prepojenie účtu s Google účtom, pridanie pretekov až po komplikovanejšiu funkcionalitu ako je pridanie výsledkov manuálne, prípadne nahraním zo súborov. Aj v tomto prípade nastali niektoré chyby, ako napríklad zle zadané číslo výkonnostného zošitu a nutnosť vyhľadať tím podľa mena psa (a prípadne psovoda), prípadne odstránenie prebytočných medzier v čísle výkonnostného zošita a iné.

9.2 Testovanie užívateľmi

Po ukončení implementácie informačného systému prebehlo aj posledné testovanie za účelom odhalenia chýb, ktoré vznikli používaním systému reálnymi používateľmi. Bola vytvorená vzorka užívateľov, ktorá sa skladala z potencionálnych užívateľov, ktorí poskytli na seba kontakt v rámci úvodného dotazníka, prípadne známych, ktorý poskytovali svoje rady pri vývoji aplikácie.

Užívateľské testovanie prebiehalo v dvoch fázach. Prvá fáza bola zameraná na samotné testovanie užívateľského rozhrania - potencionálny užívatelia mali za úlohu nájsť konkrétne preteky, prípadne konkrétnu informáciu o daných pretekoch, konkrétneho psa, alebo rozhodcu. Pripomienky užívateľov boli zapracované do výsledného užívateľského rozhrania.

Druhou fázou bolo samotné testovanie funkcionality informačného systému. Užívatelia v tejto časti vývoja dostali základnú sadu úloh, ktorých funkčnosť mali otestovať. Následne

mali ale možnosť pokračovať testovanie funkčnosti systému podľa vopred neurčeného postupu. V tejto časti testovania boli odhalené viaceré nedostatky a doporučenia ako napríklad:

- vyhľadávanie psa aj podľa čísla výkonnostného zošita;
- väčšia veľkosť mapky, ktorá sa zobrazuje u pretekov;
- možnosť pridania psa priamo pomocou tlačidla v profile, nie len pomocou rozbaľovacieho menu.

9.3 Experiment získavania asociačných pravidiel

Poslednou súčasťou testovania bolo testovanie samotnej analýzy dát. V rámci práce bolo získané väčšie množstvo dát zo samotných pretekov a ich výsledkov. Nad týmito dátami následne prebehla analýza dát pomocou asociačných pravidiel. Bola stanovená minimálna podpora 20 % a spoľahlivosť aspoň 50 %. Jednoducho povedané, v rámci experimentu boli hľadané asociačné pravidlá, kedy sa daný pes s danými vlastnosťami umiestnil na prvých piatich miestach daného behu aspoň vo 20 % prípadov a, ak má danú jednu vlastnosť, tak s pravdepodobnosťou aspoň 50 %, má daný pes aj druhú vlastnosť.

Medzi asociačné pravidlá získané z daného experimentu patrili napríklad:

- `Parson rusell teriér` $\wedge$ `S papiermi pôvodu` $\Rightarrow$ `Fenka`,
  `podpora` $= 25\,\%$, `spoľahlivosť` $= 100\,\%$
  Dané pravidlo hovorí o úspešnosti plemena Parson rusell teriér, ktorý má papiere o pôvode a je to fenka, sa v 25 % výsledkov umiestnil tento pes s danými vlastnosťami na prvých piatich miestach. Ak pes, ktorý sa umiestnil na prvých piatich miestach bol plemena Parson rusell teriér a súčastne mal papiere pôvodu, v 100 % prípadov sa jednalo o fenku.

- `Kríženec` $\Rightarrow$ `Pes` $\wedge$ `A1`,
  `podpora` $= 29\,\%$, `spoľahlivosť` $= 75\,\%$
  S pravdepodobnosťou 29 % sa pes, kríženec a zároveň vo výkonnostnej kategórii A1 (najnižšia) umiestnil na prvých piatich miestach v rámci výsledkov behov. Ak bol pes kríženec, tak s pravdepodobnosťou 75 % bol zároveň aj pohlavia pes a vo výkonnostnej kategórií A1.

- `Pes` $\wedge$ `Kríženec` $\Rightarrow$ `Bez papierov pôvodu`,
  `podpora` $= 25\,\%$, `spoľahlivosť` $= 100\,\%$
  Toto asociačné pravidlo hovorí o psoch, ktorých plemeno sa nedá jednoznačne určiť (sú krížencami), nemajú papiere pôvodu a sú to psy. Podobne ako prvé zmienené pravidlo, aj toto pravidlo má podporu 25 %, a teda, v 25 % prípadov, sa psy s vymenovanými vlastnosťami umiestnili na prvých piatich miestach vo výsledkoch behu pretekov. Následne toto pravidlo hovorí, že ak sa jednalo o psa a zároveň bol tento pes kríženec, tak v 100 % týchto prípadov daný pes nemal ani papiere pôvodu.

Tieto uvedené príklady asociačných pravidiel hovoria o pravdepodobnosti toho, že psy s danými vlastnosťami (rasa, pohlavie, papiere pôvodu a tak ďalej) sa najčastejšie umiestňujú na prvých piatich miestach. Tieto vlastnosti ale nezaručujú stopercentný úspech psa, poskytujú iba základnú analýzu výkonnosti, ktorá môže pomôcť potencionálnym pretekárom pri výbere psa. Dané pravidlá sa v budúcnosti môžu (a pravdepodobne aj budú) meniť na základe nahrávania nových výsledkov do databázy.

Kapitola 10

Záver

Cieľom tejto bakalárskej práce bolo vytvoriť informačný systém pre organizátorov agility, pričom výslednú aplikáciu je možné nasadiť aj priamo do praxe. Úlohou systému nie je iba uľahčenie organizovania oficiálnych a neoficiálnych pretekov agility, zobrazenie výsledkov pretekov, ale aj poskytnúť štatistiku výkonnosti psa a analýzu najúspešnejších plemien.

Počas celého vývoja aplikácie prebiehali konzultácie s potencionálnymi užívateľmi, aby výsledná aplikácia spĺňala všetky požiadavky pretekárov a organizátorov. Výsledná aplikácia umožňuje organizátorom pridávať preteky, výsledky jednotlivých behov uskutočnených v rámci pretekov, kontaktovať pretekárov a nahrať QR kód platby pre daného pretekára. Pretekári majú možnosť sa na preteky prihlásiť, ak spĺňajú podmienky stanovené pre účasť na pretekoch, majú možnosť kontaktovať organizátora a po uskutočnení pretekov aj pridať spätnú väzbu k organizátorovi. Užívatelia majú možnosť si zobraziť výsledky a štatistiku akéhokoľvek psa, zobraziť si jednotlivé kluby spolu s pretekmi, ktoré daný klub organizuje a rozhodcov, u ktorých majú užívatelia možnosť vidieť nadchádzajúce, prípadne uskutočnené preteky, ktoré rozhodovali. Užívateľ si môže svoj účet prepojiť s Google účtom, prípadne zobraziť Google kalendár a Google mapu, na ktorej sa nachádza vyznačené miesto konania pretekov.

Výsledné riešenie bolo otestované potencionálnymi užívateľmi a je verejne prístupné na webovej adrese `www.agility-bc.herokuapp.com`, ale plné nasadenie aplikácie je plánované na leto 2021.

Požiadavky definované pretekármi a organizátormi pred vývojom aplikácie boli naplnené, ale agility ako šport je zložitejší a dané riešenie sa môže do budúcnosti rozšíriť natoľko, aby zahrňovalo aj funkcionalitu, ktorá nezahrňuje iba organizovanie pretekov agility. Ide najmä o funkcionalitu ako je pridávanie verejných prípadne klubových tréningov, klubových akcií, rôznych seminárov a iných akcií, ktoré sú neodmysliteľnou súčasťou športu agility. V budúcnosti je takisto možné rozšíriť zobrazovanú štatistiku psa o detailnejšiu štatistiku, ako je graf rýchlosti psa a ako sa táto rýchlosť mení v závislosti na type behu, povrchu, prípadne výkonnostnej kategórie psa. Daná štatistika sa dá ďalej rozširovať keďže samotné výsledky behov poskytujú dostatok zaujímavých dát, ktoré je možné do detailov spracovávať a rozširovať pri zobrazení štatistiky psa. Okrem štatistiky psa, môže takisto v budúcnosti pribudnúť aj štatistika rozhodcov. Samozrejme, spomenuté možné rozšírenia aplikácie nie sú konečné a hlavným cieľom do budúcnosti ostáva, aby daná aplikácia napĺňala reálne potreby organizátorov a pretekárov agility.

Literatúra

[1] *Asociácia Slovenských Klubov Agility* [online]. OZ ASKA, 2011 [cit. 2021-02-09]. Dostupné z: http://www.agility.sk/.

[2] *Pravidlá Agility ASKA* [online]. Asociácia Slovenských Klubov Agility, 2020 [cit. 2021-02-09]. Dostupné z: http://www.agility.sk/dokumenty/Pravidl%C3%A1%20agility%20ASKA.pdf.

[3] *Django vs Flask – Python Frameworks Comparison* [online]. Asper Brothers, 2019 [cit. 2021-03-16]. Dostupné z: https://asperbrothers.com/blog/django-vs-flask-how-are-they-different/.

[4] *Laravel vs Symfony in 2021 – Which PHP Framework Choose For Your Project?* [online]. Asper Brothers, 2019 [cit. 2021-03-16]. Dostupné z: https://asperbrothers.com/blog/laravel-vs-symfony/.

[5] Burget, R. *Architektury informačních systémů* [online]. Brno: FIT, september 2020 [cit. 2021-04-13]. Dostupné z: https://wis.fit.vutbr.cz/FIT/st/cfs.php.cs?file=%2Fcourse%2FIIS-IT%2Flectures%2Fp02_Architektury.pdf.

[6] Divišová, K., Podešťová, M. a Benda, J. *AGILITY první krůčky.* 1. vyd. Nakladatelství PLOT, 2003. ISBN 80-86523-26-8.

[7] *Informace o webových aplikacích* [online]. Dreamweaver User Guide, 2021 [cit. 2021-03-16]. Dostupné z: https://helpx.adobe.com/cz/dreamweaver/user-guide.html/cz/dreamweaver/using/web-applications.ug.html.

[8] Foral, J. *Webová aplikace* [online]. 2013 [cit. 2021-03-16]. Dostupné z: https://wiki.knihovna.cz/index.php/Webov%C3%A1_aplikace.

[9] *Difference between MVC and MVT Design Patterns.* [online]. GeeksforGeeks, 2021 [cit. 2021-04-13]. Dostupné z: https://www.geeksforgeeks.org/difference-between-mvc-and-mvt-design-patterns/.

[10] Gillis, A. S. *What Is AWS (Amazon Web Services) and How Does It Work?* [online]. 2020 [cit. 2021-04-13]. Dostupné z: https://searchaws.techtarget.com/definition/Amazon-Web-Services.

[11] H., F. *Flask vs Django: How to Choose the Right Web Framework for Your Web App* [online]. 2018 [cit. 2021-03-16]. Dostupné z: https://blog.resellerclub.com/flask-vs-django-how-to-choose-the-right-web-framework-for-your-web-app/.

[12] *PHP Laravel VS Symfony: A Detailed Comparison of Web Development Frameworks* [online]. Hacker Noon, 2020 [cit. 2021-03-16]. Dostupné z: https://hackernoon.com/php-laravel-vs-symfony-a-detailed-comparison-of-web-development-frameworks-sq493y3l.

[13] Han, J., Pei, J. a Kamber, M. *Data Mining: Concepts and Techniques*. 3. vyd. Morgan Kaufmann, 2012. ISBN 978-0-12-381479-1.

[14] *Web framework rankings* [online]. Hotframeworks, 2021 [cit. 2021-03-16]. Dostupné z: https://hotframeworks.com/.

[15] *PostgreSQL vs MySQL: Which one should you choose?* [online]. Keboola, 2020 [cit. 2021-04-13]. Dostupné z: https://www.keboola.com/blog/postgresql-vs-mysql.

[16] Kočí, R. a Křena, B. *Uvod do softwarovérového inženýrství: 3.Prednáška* [online]. Brno: FIT, október 2020 [cit. 2021-04-13]. Dostupné z: https://wis.fit.vutbr.cz/FIT/st/cfs.php.cs?file=%2Fcourse%2FIUS-IT%2Flectures%2FIUS3.pdf.

[17] *Třívrstvá architektura (Three-tier architecture)* [online]. Management Mania, 2015 [cit. 2021-03-16]. Dostupné z: https://managementmania.com/cs/trivrstva-architektura-three-tier-architecture.

[18] *Architektura klient-server (Client–server model)* [online]. Management Mania, 2016 [cit. 2021-03-16]. Dostupné z: https://managementmania.com/cs/architektura-klient-server.

[19] Morris, S. *What Is CSS, How Does It Work and What Is It Used For?* [online]. 2021 [cit. 2021-03-16]. Dostupné z: https://skillcrush.com/blog/css/.

[20] *JQuery* [online]. Mozilla, 2020 [cit. 2021-03-16]. Dostupné z: https://developer.mozilla.org/en-US/docs/Glossary/jQuery.

[21] *About JavaScript* [online]. Mozilla, 2021 [cit. 2021-03-16]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/JavaScript/About_JavaScript.

[22] *Django introduction* [online]. Mozilla, 2021 [cit. 2021-04-13]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Introduction.

[23] *13 Best JavaScript Frameworks For 2020* [online]. Nikhil, 2020 [cit. 2021-03-16]. Dostupné z: https://www.lambdatest.com/blog/best-javascript-framework-2020/.

[24] *WebFrameworks* [online]. Python Wiki, 2021 [cit. 2021-04-13]. Dostupné z: https://wiki.python.org/moin/WebFrameworks.

[25] *What is Python? Executive Summary* [online]. Python.org, 2021 [cit. 2021-04-13]. Dostupné z: https://www.python.org/doc/essays/blurb/.

[26] *Cascading Style Sheet (CSS)* [online]. Technopedia, 2021 [cit. 2021-03-16]. Dostupné z: https://www.techopedia.com/definition/26268/cascading-style-sheet-css.

[27] *Best CSS Frameworks in 2021* [online]. ThemeSelection, 2020 [cit. 2021-03-16]. Dostupné z: https://dev.to/theme_selection/best-css-frameworks-in-2020-1jjh.

[28] *MVC Framework - Introduction* [online]. Tutorialspoint, 2021 [cit. 2021-04-13]. Dostupné z: https://www.tutorialspoint.com/mvc_framework/mvc_framework_introduction.htm.

[29] *Heroku* [online]. VIP Trust, 2021 [cit. 2021-04-13]. Dostupné z: https://viptrust.com/technologie/ostatni/heroku.

[30] *Introduction to HTML* [online]. W3schools, 2021 [cit. 2021-03-16]. Dostupné z: https://www.w3schools.com/html/html_intro.asp.

[31] Zendulka, J., Bartík, V., Lukáš, R. a Rudolfová, I. *Získávání znalostí z databází* [online]. Brno: FIT, október 2009 [cit. 2021-04-14]. Dostupné z: https://wis.fit.vutbr.cz/FIT/st/cfs.php.cs?file=%2Fcourse%2FZZN-IT%2Ftexts%2FZZN.pdf.

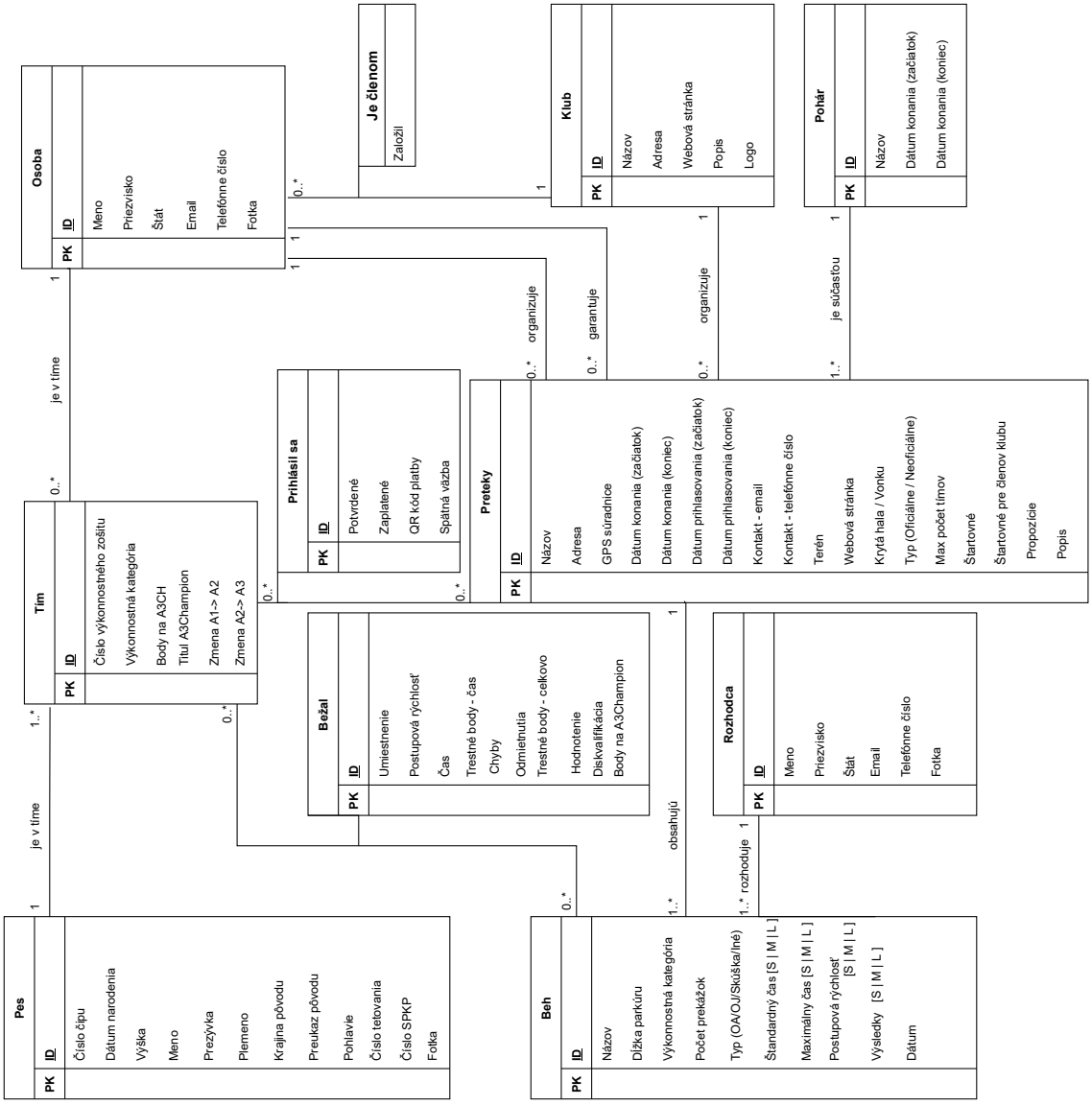
Príloha A

Obsah priloženého pamäťového média

- **app-src** - zdrojové súbory aplikácie.
- **readme.txt** - súbor popisujúci štruktúru pamäťového média
- **sprava-src** - zdrojové súbory technickej správy
- **xondru16.pdf** - technická správa

Príloha B

ER Diagram



Obr. B.1: Celkový ER diagram